Universidade do Algarve
Faculdade de Ciências e Tecnologia

Substation Automation Systems and IEC 61850:
Interoperability Testing

Henrique A. Guerreiro Dias Evaristo

Master in Electronics and Telecommunications Engineering
2011

Universidade do Algarve
Faculdade de Ciências e Tecnologia

Substation Automation Systems and IEC 61850:
Interoperability Testing

Henrique A. Guerreiro Dias Evaristo

Supervisors:

Carmo Medeiros
Alvaro Barradas
Bas Mulder

Master in Electronics and Telecommunications Engineering
2011

ABSTRACT

The Substation Automation System (SAS) is the backbone of the Energy Power System (EPS) and IEC 61850 is becoming its single most important standard. This is a world wide accepted standard that is being adopted by the industry in order to provide for all current and future needs. This standard defines not only the communication protocols but also its own Substation Configuration Language (SCL) and even best practices for related engineering processes. In order to keep up with the current fast technological developments the substation data model is separated from the communication protocols allowing both to be changed without affecting each other. Also defined in the standard is an extensive conformance testing procedure in order to guarantee that different vendors interpret and implement the standard correctly. The substation and its SAS must undergo thourough testing procedures specificaly in the Factory Acceptance Test (FAT) and Site Acceptance Test (SAT). The conformance tests insures that the SAS devices, the Intelligent Electronic Devices (IEDs), conform to the same standard but on its own does not guarantee its interoperability. An automated testing tool capable of, quickly and easily, testing the SAS functions (IEDs interoperability) provides significant savings in both time and money to the testing process. This work aim is to develop such a tool, capable of interoperability testing. In order to achieve such big accomplishment this initial work focus on only two of the most used functions of the SAS: switching and interlocking. A simulation model, built on top of OMNeT++, for both the IEDs and the substation was developed. In this work an initial stage prototype with an IED simulation model capable of communicating with real devices will be developed. In a later stage, postponed for future work, the substation simulation model will be extended in order to include real-time interaction with external devices that emulate the substation switchgear.

Keywords: Substation Automation System, IED, Simulator, 61850, Interoperability

RESUMO

Os Sistemas de Automação de Subestações (SAS) são actualmente uma parte essencial da engenharia de subestações. Neste âmbito o *standard* internacional IEC 61850 começa a ganhar extrema relevância, tendo sido aceite e, lentamente, adoptado por todo o mundo. Este *standard* define não apenas os protocolos de comunicações mas também os processos de engenharia e uma linguagem de configuração da subestação. O modelo de dados encontra-se separado dos protocolos de comunicação usados o que permite, a cada um, acompanhar os mais recentes desenvolvimentos da tecnologia sem se afectarem. Também descrito no *standard* está um extenso programa de testes de modo a garantir a conformidade entre dispositivos de diferentes fornecedores. O programa de testes é uma parte importante do processo de engenharia de uma subestação. Neste contexto a conformidade apenas, não garante, por exemplo, a interoperabilidade entre dispositivos. Isto faz com que um esforço, maior do que o necessário, seja despendido no teste de funcionalidade do SAS e dos seus dispositivos antes do comissionamento final.

Este trabalho pretende desenvolver uma aplicação de testes das funcionalidades da SAS, testando a interoperabilidade entre dispositivos, que permita uma redução significativa dos custos associados aos processos de teste. Para tal foi desenvolvido um modelo de simulação de subestações que permite a interacção com dispositivos reais e que, de forma automatizada, reduza o tempo despendido. Tendo em conta a dimensão de tal tarefa, o trabalho assume à partida que todos os objectivos relacionados não poderão ser atingidos dentro do limite de tempo estipulado. Como tal este trabalho irá focar-se numa primeira fase apenas nas funções de comutação (*switching*) e inter-bloqueio (*interlocking*) com o desenvolvimento do modelo de simulação da subestação e do modelo de simulação dos dispositivos do SAS incluindo a interacção com dispositivos reais. Numa segunda fase, no futuro, a interacção com dispositivos reais do modelo de simulação da subestação será adicionada bem como outras funções da subestação.

Keywords: Sistemas de Automação de Subestações, IED, Simulador, 61850, Interoperabilidade

ACKNOWLEDGEMENTS

I would like to start by thanking KEMA for the opportunity given and for the thrust they deposited into me for the development of this work. In particular I would like to thank Maurice Adriaensen for he was the responsible to introduce me to KEMA and also for this project to become a reality. I would also like to thank Bas Mulder, my supervisor, and Richard Schimmel for all the support and knowledge they so willingly provided and shared with me. Without their expertise and experience such work would have been soo much harder and possibly impossible to achieve. I would also like to thank Robin Massink, Pierfrancesco Cioci and Gerard Akse for all the patience and help they also provided even though it was not part of their responsibility. Also a special thanks goes to all the remaining office colleagues that directly or indirectly contributed to this work and also to my extremely pleasant stay in the Netherlands.

This study and a big part of my education is also the responsibility of Universidade do Algarve. I would like to thank to my professors there that throughout the years helped, educated and served me as my inspiration throughout the years. In particular I would like to thank my supervisor, Carmo Medeiros, for she is also the direct responsible for this work. Even during busy times she always tried to support and guide me to a positive outcome. I would also like to thank Alvaro Barradas for he was a big inspiration and also a big supporter, particularly in the simulation field that he so much understands. A special thank also goes to my professor A. Isabel Leiria that even during difficult times manages to stand and reach out to students when they most need it. I would also like to mention and appreciate the work of other professors like Margarida Madeira e Moura, Sérgio Jesus, António Casimiro, António Ruano, M. Graça Ruano and José Valente de Oliveira. They all contributed positively to my education. Also important was the support and friendship from my colleagues, too many to name but hopefully they know their importance to. They made my time there enjoyable and fun. Also thanks to the student union, NEI, that many times provided for the installations, material and environment to study and work.

Finaly but not least my thanks goes to my family. My mother, Conceição Evaristo, and my father, Henrique José Evaristo, that always provided me with everything I needed and could hoped for. My special thanks also goes to my godmother Celeste Ildefonso. I also need to thank the remaining of my, huge, family for they are an important part of my life as well. My grandparents, uncles, aunts and cousins. Special thanks also to Laura, my girlfriend, for everything she is and also does for me. You are all, in some way, responsible for the person I am today and everything I achieved so far. Thank you all for all your love and support.

“nanos gigantium humeris insidentes”

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF LISTINGS

Page

ACRONYMS

AC	Alternating Current
ACSI	Abstract Communication Service Interface
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BASIC	Beginner's All-purpose Symbolic Instruction Code
CC	Control Center
CDC	Common Data Class
DLL	Dynamic Link Library
EPS	Energy Power System
F	Function
FAT	Factory Acceptance Test
GOOSE	Generic Object Oriented Substation Events
GUI	Graphical User Interface
HMI	Human Machine Interface
ICMP	Internet Control Message Protocol
IEC	International Electro-technical Commission
IED	Intelligent Electronic Device
IID	Instantiated IED Description
IP	Internet Protocol
IS	International Standard
ISO	International Standards Organization
ITU	International Telecommunication Union
KITS	KEMA IEC Test System
LC	Logical Connection
LD	Logical Device
LN	Logical Node
NCC	Network Control Center
MMS	Manufacturing Message Specification
MOC	Management Operational Consulting
OO	Object-Oriented
OOD	Object-Oriented Design
OS	Operating System
OSI	Open Systems Interconnection
PC	Physical Connection

PD	Physical Device
PDH	Plesiochronous Digital Hierarchy
PID	Project Initiation Document
PIXIT	Protocol Implementation eXtra Information for Testing
PCTC	Protocol Competence & Test Center
RCC	Regional Control Center
RFC	Request For Comments
RTU	Remote Terminal Unit
SA	Substation Automation
SAS	Substation Automation System
SAV	Sampled Analog Values
SAT	Site Acceptance Test
SBO	Select Before Operate
SCADA	Supervisory Control And Data Acquisition
SCD	Substation Configuration Description
SCL	Substation Configuration Language
SCSM	Specific Communication Service Mapping
SDH	Synchronous Digital Hierarchy
SEP	System Engineering Process
SONET	Synchronous Optical Networking
SSD	System Specification Description
SUT	System Under Test
TC57	Technical Committee 57
TCL	Tool Command Language
TCP	Transmission Control Protocol
TTL	Time To Live
UCA	Utility Communication Architecture
UML	Unified Modeling Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Scope

This study focus on Substation Automation Systems (SASs) particularly in the communication networks and systems and the conformance and interoperability testing of the devices involved in SAS. Currently International Electro-technical Commission (IEC) standard 61850 is becoming, if not already, the *de facto* standard for SAS and for that reason this study looks at it exclusively. The work presented here is performed in cooperation with KEMA Protocol Competence & Test Center located in Arnhem, Netherlands. The aim is to design and develop a software application, capable of automated conformance and interoperability testing on multiple Intelligent Electronic Devices (IEDs) by means of simulation.

1.2 Motivation

With the introduction of commercial communication technologies in the SAS many new possibilities were opened for substation automation engineering such as the adoption of Ethernet and open cost-effective solutions for communications. The recent deregulation of the energy market bringing new strict economic requirements on the utilities processes. The growing number of new methods for energy production, wind, solar and others, bringing new challenges to the electrical grid, like variable availability (what to do when the wind stops or changes intensity). These are some of the reasons that both the utilities and SAS providers have to consider in order to improve the engineering processes. An electrical grid capable to accommodate all these requirements, predicting and intelligently responding to different users needs is what is commonly called smart grid. SAS equipment evolved from the simple Remote Terminal Unit (RTU) to the IED. Several components and devices from different manufacturers have to inter operate flawlessly and communication standards are a basic requirement to realize open infrastructures and device interoperability. IEC 61850 standard attempts to capture all these requirements by providing an object model, it's data and services and also the engineering support. All of these on top of mainstream, open communication standards. Allowing multi-vendor device interoperability while at the same time enabling enough room for different vendors to employ different strategies and philosophies. The conformance and interoperability testing are important steps in the substation life cycle and as such several parts of the standard are dedicated to them.

The processes of constructing a new substation, substation expansion and/or substation re-

furbishment are common among electrical utilities and require large amounts of capital. With the advent of deregulation, the utilities need to reduce the substation costs and the optimization of the grid usage became larger. This means that a substation outage due to construction, expansion and/or refurbishment needs to be reduced to a minimum. Site Acceptance Test (SAT) and problem solving increases the scheduled outage time, as such time spent in Factory Acceptance Test (FAT) must be optimized and the tests must be improved. Automation and improvement of the testing procedures makes the FAT process more efficient and reliable thus reducing the time spent on SAT. Since substations life cycle are an important and expensive part of the utilities processes. Substations having a life expectancy of 50/60 years before replacement while it's secondary equipment is replaced every 5-10 years. Enabling more efficient processes to be employed results in significant cost reductions allowing utilities to improve their competitiveness in the new free, deregulated, energy market.

Many new tools have been created to be used and to help in these new processes but there is still demand for further development and improvement. Tools such as protocol analyzers, simulators, Substation Configuration Language (SCL) checkers, Generic Object Oriented Substation Events (GOOSE) simulators are already developed and under constant improvement and are extensively used in the industry. The standard goal is indeed to achieve device interoperability but such a goal cannot currently be guaranteed when only the standard conformance is observed. Many tools exists but the testing process still relies heavily on manual error prone work and its main focus is on the standard conformance only. Advantages and benefits can be expected by developing a tool capable of testing the multi device interoperability capabilities.

1.3 KEMA

Established in 1927, KEMA is a global provider of high-quality services to the energy value chain, including business & technical consultancy, operational support, measurements & inspection, and testing & certification. Besides certifying products, processes and individuals, KEMA advises governmental bodies as well as companies and institutions involved with energy. Within KEMA, the Protocol Competence & Test Center provides tools, testing, certification and consulting services for standard IEC 61850 and others.

1.4 Goals and contributions

The ultimate aim of this study is to design and develop a tool capable of performing interoperability testing of multiple IEDs made by different vendors. In order to test and completely perform interoperability between devices the tool must totally understand the substation envi-

ronment and its constituent elements. As such it is necessary to provide a complete simulation model of both primary and secondary equipment involved in SAS. Such a tool is then able to test all of the different functions of the SAS. This work objectives assumes that the resources available will not allow the complete achievement of this goal and therefore a small subset of the total functions of the SAS are chosen. These will be the switching and interlocking functions. Due to the current and future importance of IEC 61850 standard for SAS this work will be limited to it.

Switching is one of the main functions of the substation and definitely one of the most used. In conjunction with switching, interlocking is also an important function. Interlocking provides the security features to, e.g, keep the human operators safety safeguarded inside the substation. The testing procedures are simple and often thousands of them are required to be performed in a single substation. Automation of these tests favor from several factors. Simple, repetitive tasks have a high probability of errors introduced by human operators. Simple testing procedures are easily automated. Automation reduces the time and money spent on testing.

This work contribution will be an important step in the main aim by providing valuable insight into future requirements, constraints and possible strategies to follow in the pursuit of a complete commercial ready automated tool for interoperability testing.

1.5 Document structure

This dissertation document is divided into different chapters and sections briefly described in the following paragraphs.

The current chapter, chapter 1, contains the introduction to the work, it's scope, motivation, introduces KEMA and also details the intended project goals and contributions. The literature review and *state of the art* is exposed on chapters 2, 3 and 4. Brief overviews of the topics under discussion are presented. The Energy Power System (EPS) in chapter 2 is explained in order to get into context and the historical perspective is presented to understand the current situation. The IEC and IEC 61850 standard is presented in chapter 3. The simulation models and platforms are discussed on chapter 4. In chapter 5 the work development is discussed with the different decisions that had to be made and the reasoning why. The results obtained are exposed in chapter 6 and the conclusions in chapter 7. Also in the conclusions chapter a section on future work is presented. This is an important part since the project assumed only a small subset of the complete SAS functions and assumes future work will be developed based on the provided foundations.

Chapter 2

Electrical Power System

2.1 Power production

Current Human society depends heavily on the Energy Power System (EPS) and on the availability of electric power. This electrical energy is produced at industrial facilities that have the common designation of power plants or power stations.

Most of the power plants in use today produce electric power by rotating one or more generators that transform the mechanical energy into electrical energy, using the relative motion between a magnetic field and a conductor (electromagnetic induction). The main energy sources for the mechanical movement have always been fossil fuels, nuclear and hydro-power.

Electric energy produced by a generator using the electromagnetic induction principle results in a sinusoidal, Alternating Current (AC), electric signal. Displacing three poles equally across the generator allows three electric conductors with AC phase difference between them of 120°. The three phase system presents several desirable properties and advantages that made it into a common method for transferring power worldwide.

The first power plant started operation in 1882 by the hands of Thomas Edison and Edward Johnson, a boiler powered 125 horsepower steam engine that drove a 27 ton generator.

Initial power plants were relatively close to their customers and served a small number of users, this allowed for a simple and primitive system of distribution. As soon as the number of customers increased this was no longer possible as few power production centers had to provide for many consumers and a system to transport the electric power over long distances efficiently was necessary.

2.2 Power distribution and transmission

The electric power transmission and the electrical power distribution are distinct from each other in the sense that transmission refers to the transfer of electrical energy from generation centers to high voltage substations near population centers while the local wiring between high voltage substations and customers is considered the distribution. The goal of the transmission system to transmit electric power efficiently is achieved by transforming the generated voltages into higher voltage levels. Since lower voltages are easier and safer to handle the distribution system needs to transform the higher voltages coming from the transmission system back into lower voltages.

Historically, because of the few initial consumers, the differences between the transmission and the distribution systems were non-existent or too small and they were owned by the same company.

A simple representation of the complete electrical power system is presented in figure 2.1 with the generation, transmission and distribution systems along with several possible customers.

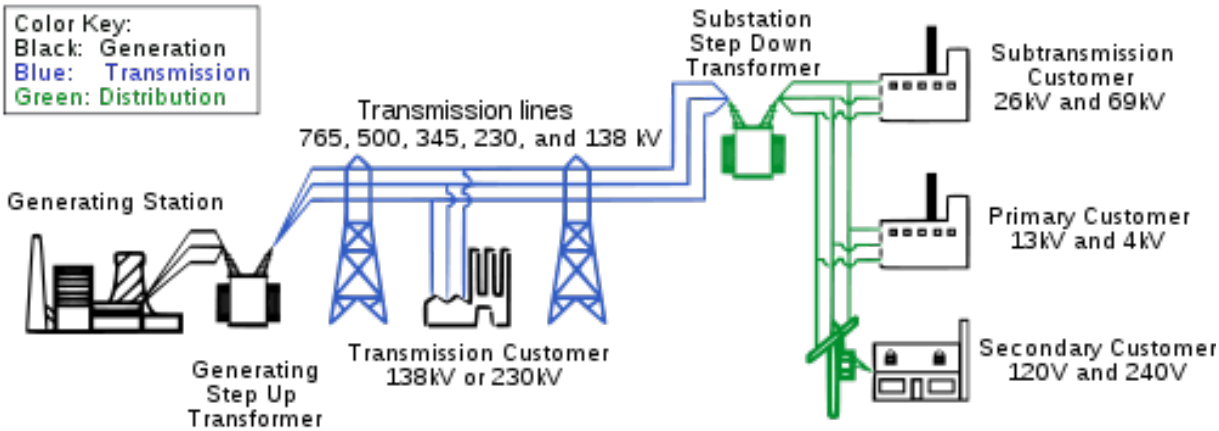

Figure 2.1
Electrical power system diagram (Wikipedia (2011)).

2.3 Electrical substations

In the early days of the EPS, the power plant was connected to a single station that was its subsidiary, acquiring the common designation of substation. In current power networks, the substation is the linking node connecting the generation, transmission and distribution systems. Inside it the busbar is the main connection element and with specific equipment it is able to connect incoming and outgoing lines. The substation itself is subdivided into what is called a bay. Usually if the bay is used to connect a power line to the busbar system, it is called a line bay, if it is used for connecting a power transformer to the busbar system, it is called a transformer bay, etc. Depending on the substation functions different common structures and circuit configurations have been devised.

Substation equipment is mainly divided into primary equipment, related to the main functions of the substation, and secondary equipment for control, protection and monitoring of the primary equipment. Primary equipment include among others:

- Circuit Breakers;

- Disconnectors;
- Earthing switches;
- Instrument transformers;
- Transformers.

In order to represent a three phase power system a simplified notation, one-line or single-line diagram, is used where electrical elements such as circuit breakers, disconnectors, transformers and conductors have standardized schematic symbols. The main advantage comes from the fact that instead of representing each of the three phases conductors with distinct lines, only one conductor is shown in the diagram. The need to maintain the power load on each phase balanced allows this simplification.

Two common substation configuration circuits are shown in figures 2.2 and 2.3, commonly named the single busbar and the double busbar respectively. In the single-line diagram the busbar is usually represented by a long horizontal line, the lines numbered I and II in figure 2.3, the disconnector as an unconnected oblique line and the circuit breaker like a disconnector but with a cross.

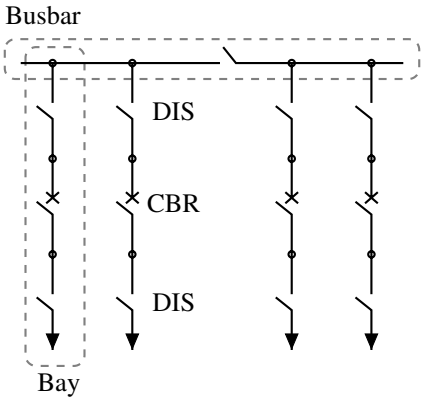

Figure 2.2
Single-line diagram for a single busbar substation configuration. This particular substation contains 4 bays, each with 2 disconnectors (DIS) and 1 circuit breakers (CBR).

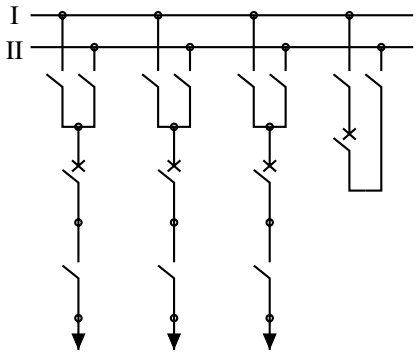

Figure 2.3
Single-line diagram for a double busbar substation configuration. This particular substation contains 4 bays, 3 connected to feeders and another one for busbar selection.

Mainly four different types of substations can be found in the power network. The switch-yard, that connects the generation plants to the network. The customer substation, that provides energy power to one particular business customer. The switching substation, that typically serve as endpoints for the transmission lines. The distribution substation, that supply the end users.

2.3.1 Power network Management System

In order to control, monitor and protect the power network a management system was created. The main purpose of the system is to have direct control of the network, control the energy flow path, maintain the balance between produced and consumed energy, business related functions and energy quality and availability. The substation is the lowest level of this system contained inside regions, each managed by a Regional Control Center (RCC), under the Network Control Center (NCC) top level. Different functions must be performed at the different levels of the management network. These tasks mainly involve the acquisition of data and the control of the system and is generally referred to as Supervisory Control And Data Acquisition (SCADA).

The power network management system is a distributed system where its lowest level is the substation and its automation systems. The specific functional and performance requirements in conjunction with the long lifetime of the power equipment impose several constraints that must be taken into consideration. A possible representation for the NCC hierarchy levels is shown in figure 2.4.

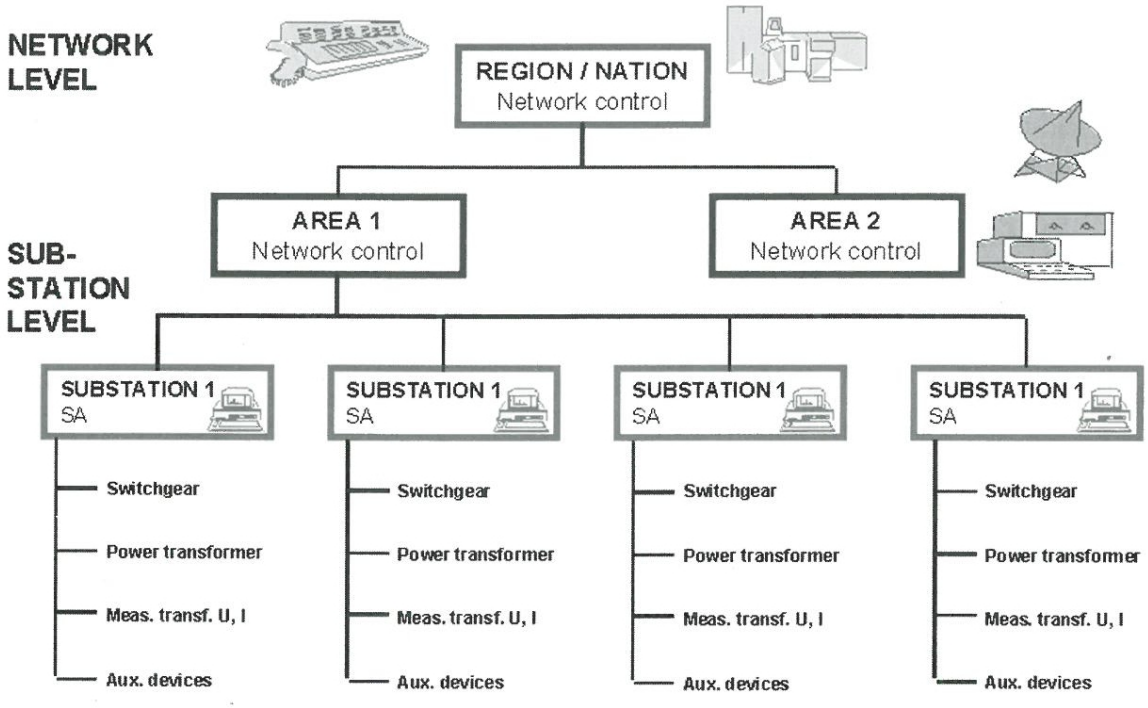

Figure 2.4
Network Control Center hierarchy representation (Brand et al. (2003)).

2.3.2 Automation System

Initially there was no Substation Automation Systems (SASs) but simple Remote Terminal Units (RTUs) placed in each substation. These initial systems had few local functions and mainly existed to serve the NCC with information. Together they formed the SCADA system. With time, systems to perform local related functions in a decentralized way began to appear. These systems could provide local and remote access to the substation power system automating some of the local tasks like data collection and storage. Many Intelligent Electronic Devices (IEDs) to perform such functions began to appear and they could be performed by a single device or by many different devices communicating with each other. The move from electromechanical to fully digital electronic technology in the SAS allowed the average outage time to change from 2 days to 10 minutes per year.

Initial communication systems between the RTU and the NCC had a narrow bandwidth of 20 bit/s up to 2.4 kbit/s. Automation was primitive and could easily be monitored by the operating staff. Advances like optical cables, Synchronous Optical Networking (SONET), Plesiochronous Digital Hierarchy (PDH), Synchronous Digital Hierarchy (SDH), Gigabit Ethernet allow today bandwidths of 10 Mbit/s up to 10 Gbit/s. It is now possible to employ wide area protection schemes that operate globally in less than 0.5 seconds. These schemes involve satellite time synchronization and communication, broadband communication networks and SAS and can ultimately prevent the spreading of faults throughout the network.

With the introduction of digital SASs, communication between IEDs within the substation became a primary performance issue. Different vendors devised their own proprietary protocols. Complicated and costly protocol converters when using IEDs from different vendors became a necessity. This results in complex and difficult Substation Automation (SA) maintenance process. The need for standard communication protocols and the technology advancements paved the way for standardization.

2.4 Energy market changes

The heavy dependence of society on the energy service have made it to be considered vital and of public interest and for that it has always been regulated by the different governments. Only recently, with the intent to stimulate markets to reduce power costs and increase power products and services, the energy market has been deregulated. This deregulation resulted in a change in the industry structure and how the consumer purchases its energy, allowing him the chance to choose its provider. A bigger separation and a split between the generation, transmission and

distribution systems among different companies was necessary as different companies exploited different strategies in order to become more competitive in the new open market.

In the recent years, with the global awareness for the need to renewable energy sources and the improvements in technology and research funding, the number of wind farms and photovoltaic cells have been increasing. Their market share is still low but steadily increasing.

2.5 Challenges and the future

Society requirements for an ever-increasing energy demand and a sustainable balance between economic interests, affordable energy, and growing concerns about the environment. The market deregulation and the energy utilities restructuring. The new emergent micro-generation (local energy) communities, presenting small but many new generation centers. The new generation methods that present different availability challenges, like wind and solar farms.

Chapter 3

IEC 61850 International Standard

3.1 International Electrotechnical Commission Organization

The International Electro-technical Commission (IEC) is the world's leading organization that prepares and publishes International Standards (ISs) for all electrical, electronic and related technologies. It is a not-for-profit, non governmental organization, founded in 1906. Over 10000 experts from industry, commerce, government, test and research labs, academia and consumer groups participate in IEC Standardization work. It's members are National Committees, they appoint experts and delegates coming from industry, government bodies, associations and academia to participate in the technical and conformity assessment work of the organization. IEC is one of three global sister organizations (IEC, International Standards Organization (ISO), International Telecommunication Union (ITU)) that develop International Standards for the world. When appropriate, IEC cooperates with ISO or ITU to ensure that International Standards fit together seamlessly and complement each other. Joint committees ensure that International Standards combine all relevant knowledge of experts working in related areas.

3.2 IEC 61850 standard

IEC 61850 standard was released between 2003 and 2005 while its second edition formal release came in 2011. The move from electro-mechanical devices to digital devices for Substation Automation System (SAS) equipment, resulted in the development of specific proprietary communication protocols by different manufacturers. The industry's needs and opportunity to develop standard communication protocols allowing interoperability between Intelligent Electronic Devices (IEDs) of different manufacturers propelled the development of this standard. The standard is divided into several parts under the general title Communication networks and systems in substations.

- Part 1: Introduction and overview
- Part 2: Glossary
- Part 3: General requirements
- Part 4: System and project management
- Part 5: Communication requirements for functions and device models
- Part 6: Configuration description language for communication in electrical substations related to IEDs

- Part 7-1: Basic communication structure for substation and feeder equipment – Principles and models
- Part 7-2: Basic communication structure for substation and feeder equipment – Abstract Communication Service Interface (ACSI)
- Part 7-3: Basic communication structure for substation and feeder equipment – Common Data Class (CDC)
- Part 7-4: Basic communication structure for substation and feeder equipment – Compatible logical node classes and data classes
- Part 8-1: Specific Communication Service Mapping (SCSM) – Mappings to Manufacturing Message Specification (MMS) and to ISO 8802-3 (2000)
- Part 9-1: SCSM – Sampled values over serial unidirectional multidrop point to point link
- Part 9-2: SCSM – Sampled values over ISO 8802-3 (2000)
- Part 10: Conformance testing

Part IEC 61850-9-1 (2003) has been proposed for withdrawal in 2009 and since discontinued.

As stated in IEC 61850-1 (2003), the objective of Substation Automation (SA) standardization is to develop a communication standard that will meet functional and performance requirements, while supporting future technological developments. According to the standard, functions are tasks, which are performed by the substation automation system, i.e. by application functions. Generally, functions exchange data with other functions. The details are dependent on the functions in consideration. Functions are performed by IEDs (physical devices). Functions may be split in parts residing in different IEDs but communicating which each other (distributed function) and with parts of other functions. These communicating function parts are called Logical Nodes (LNs). In the context of this standard, the decomposition of functions or their granularity is ruled by the communication behavior only. Therefore, all functions considered consist of logical nodes that exchange data. The standard makes use of Object-Oriented Design (OOD) techniques as such, the reader requires a clear understanding of classes and instances.

IEC 61850-6 (2004) defines a configuration description language to be used in the intended new substation engineering process. The requirements for mutual understanding of devices from different suppliers results in a data and communication service model defined in parts 7-x. The mapping of this model to communication stacks is described in parts 8-x and 9-x. By employing OOD techniques in its specification, the standard ensures its technological independence and allows different utilities philosophies, requirements and constraints.

Figure 3.1 shows the overall communication system architecture and how the different parts of the standard are modeled and implemented.

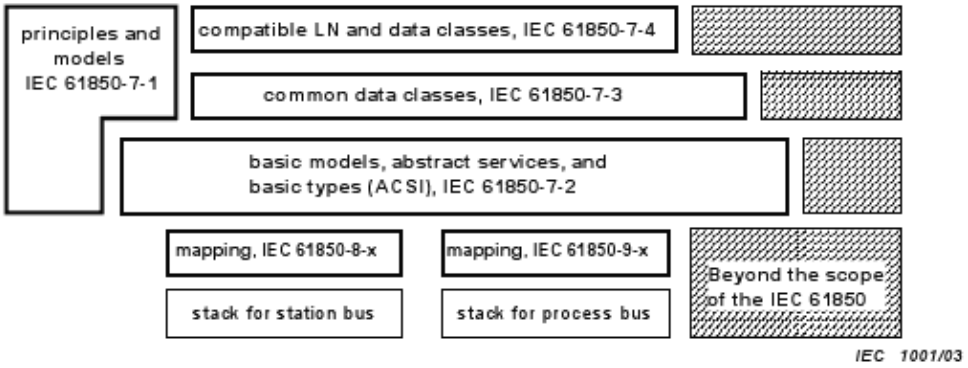

Figure 3.1
IEC 61850 overall communication system architecture (IEC 61850-5 (2003)).

3.3 Concepts

The following are some of the most important concepts to understand about the standard. A Function (F) is decomposed into several LNs. LNs may reside in one or more Physical Device (PD). LN are linked by Logical Connection (LC) and the PD by Physical Connection (PC). These concepts are represented graphically in figure 3.2 where two different distributed func-

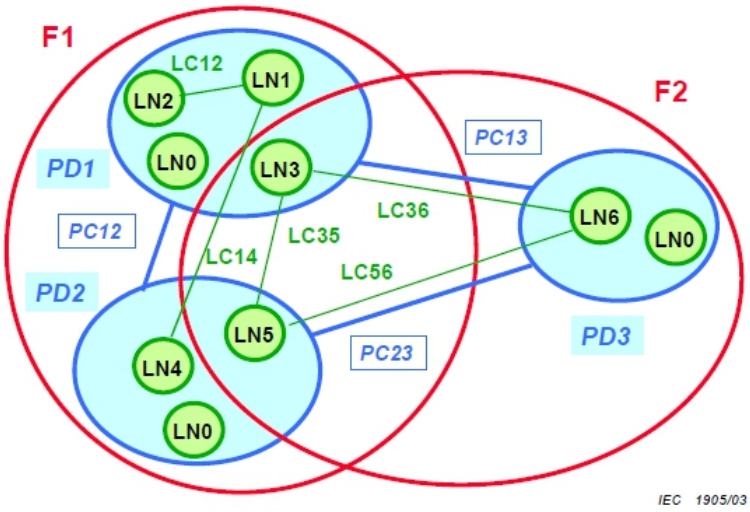

Figure 3.2
IEC 61850 Logical node and link concepts (IEC 61850-5 (2003)).

tions, *F1* and *F2*, are represented. They are considered distributed, since the composing LNs are located in different physical devices, i.e*PD1* and *PD2* for *F1*.

The functions of a SAS refer to tasks, which have to be performed in the substation. These

include control, monitor and protection but since the standard intends to describe also the engineering process they also include system configuration, communication management or software management. The standard describes the substation environment at three distinct levels with different interfaces between them. In figure 3.3 one can observe the three levels, process, bay/unit and station, and the different logical interfaces that connect them, 10 in total.

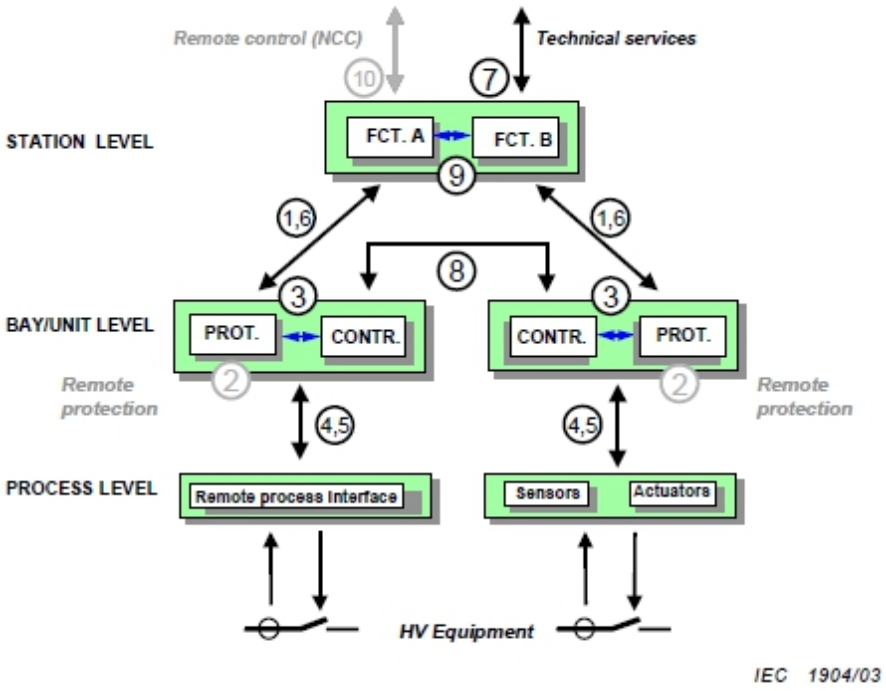

Figure 3.3
IEC 61850 levels and logical interfaces for substation automation systems (IEC 61850-5 (2003)).

3.4 Substation Configuration Language

According to the standard the Substation Configuration Language (SCL) in its full scope describes a model of:

- The substation primary (power) system structure: which primary apparatus functions are used, and how the apparatus are connected. This results in a designation of all covered switchgear as substation automation functions, structured according to IEC 61346-1 (1996) (currently replaced by IEC 81346-1 (2009)).
- The substation communication system: how IEDs are connected to subnetworks and networks, and at which of their communication access points (communication ports).

- The application level communication: how data is grouped into data sets for sending, how IEDs trigger the sending and which service they choose, which input data from other IEDs is needed.
- Each IED: the logical devices configured on the IED, the logical nodes with class and type belonging to each logical device, the reports and their data contents, the (pre-configured) associations available; and which data shall be logged.
- Instantiable LN type definitions. The LN as defined in IEC 61850-7-x have mandatory, optional and user defined DATA (here abbreviated DO) as well as optional services, and are therefore not instantiable. In this document, instantiable LNTypes and DOTypes are defined as templates, which contain the really implemented DOs and services.
- The relations between instantiated logical nodes and their hosting IEDs on one side and the switchyard (function) parts on the other side.

The intention of the SCL is to exchange the configuration data between different tools, possibly from different manufacturers. There at least four different purposes for SCL data exchange and has such four different kinds of SCL file types exist, identifiable by different file extensions. The SCL file relevant for this study contains all IEDs, a communication configuration section and a substation description section. The file extension used is ".SCD" from Substation Configuration Description (SCD).

SCL is based on Extensible Markup Language (XML) and it's syntax definition is described as a World Wide Web Consortium (W3C) schema definition. Among others the SCL schema structure contains one SCL element containing several other elements like *Substation*, *IED* and or *Communication*. Depending on the type of SCL file, some of these elements may not be necessary and as such not present. For the type of SCL required by this study, all of the mentioned elements must be present with at least one but also possibly more instances.

3.4.1 Substation description

Quoting the standard, still referring to SCL:

> The substation section serves to describe the functional structure of a substation, and to identify the primary devices and their electrical connections. For an industrial process or to describe whole power networks, it is possible to have several substation sections, one for each substation served by the SAS.

Some of the elements used in this section are, ordered by hierarchy, the *Substation*, *VoltageLevel*, *Bay*, *Equipment*, *SubEquipment* and *Terminal*. The *ConnectivityNode* found under the *Bay* element connect different primary devices. The *PowerTransformer* is a special equipment, which can hierarchically be located below *Substation*, *VoltageLevel* or *Bay*. It contains

Transformer windings as equipment, which might again have a relation to a tap changer. Elements relating logical nodes to a specific substation part, equipment or sub-equipment are found hierarchically bellow the related element, e.g. a LN of the type XCBR can be found bellow the *Equipment* or the *SubEquipment* element depending on the modeling of a single phase or three-phase equipment.

3.4.2 IED description

The scope of SCL restricts the product side to cover the hardware devices (IEDs) that form the substation automation system. As such only these are described in SCL. Quoting the standard definition:

> The IED section describes the (pre-)configuration of an IED: its access points, the logical devices, and logical nodes instantiated on it. Furthermore, it defines the capabilities of an IED in terms of communication services offered and, together with its LNType, instantiated data (DO) and its default or configuration values. There shall be one IED section for each IED. IED names (name attribute) shall be unique within the file.

As such some of the elements that compose it are the *IED*, *AccessPoint*, *Server*, *LDevice* and *LNode*.

3.4.3 Communication system description

The definition quoted from the standard is:

> This Clause describes the direct communication connection possibilities between logical nodes by means of logical buses (SubNetworks) and IED access points. The IED sections already describe which LDs and LNs are reachable across a certain access point. The communication section now describes which IED access points are connected to a common subnetwork. This is done in a way that reflects the hierarchical name structure within the IED, which is based on IED relative names for access points, LDs and LNs.

The elements composing the communication system are the *Subnetwork*, *ConnectedAP* and the *Address*.

3.5 Logical Nodes

According to the standard most of the functions consist of a minimum of three logical nodes, i.e. the LN with the core functionality itself, the process interface LN and the Human Machine

Interface (HMI). In IEC 61850-5 (2003) part of the standard, section 11, one can find the list of LN grouped by core functionality. For this study the LNs with the most relevance are listed in tables 3.1 and 3.2. These include the control and primary equipment LNs.

Table 3.1
IEC 61850 Logical Nodes (LNs) for control.

Logical Node	Acronyms	Description
Switch controller	CSWI	The switch control LN handles all switchgear operations from the operators and from related automatics. It checks the authorization of the commands. It supervises the command execution and gives an alarm in case of an improper ending of the command. It asks for releases from interlocking, synchrocheck, autoreclosure, etc. if applicable.
Interlocking function	CILO	

Table 3.2
IEC 61850 Logical Nodes (LNs) related to primary equipment

Logical Node	Acronyms	Description
circuit breaker	XCBR	An AC circuit breaker is a device that is used to close and interrupt an AC power circuit under normal, fault or emergency conditions.
switch	XSWI	Line switch is a switch used as a disconnecting, load-interrupter, or isolating switch on an AC or DC power circuit.

The interactions between LNs are not clearly defined or enforced by the standard. This allows the freedom to different vendors to exploit different strategies and philosophies but also brings extra difficulties and complexities. Figure 3.4 presents an example implementation for several functions, including interlocking.

3.5.1 Data and Data Attributes

The LN contains DATA classes that are used in the definition of almost all necessary information. The data class, defined in IEC 61850-7-2 (2003), is very generic and therefore it must be specified/specialized by the definitions of common data classes specified in IEC 61850-7-3 (2003). These common data classes are later used by many more definitions specified in IEC

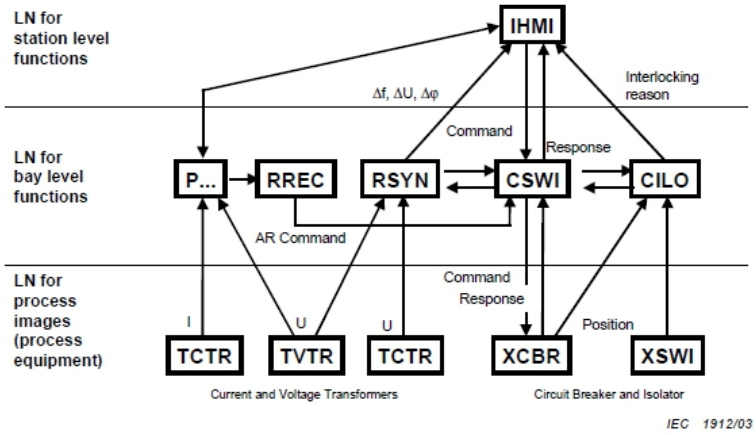

Figure 3.4
IEC 61850 Example for interaction of LNs for switchgear control, interlocking, synchrocheck, autoreclosure and protection (IEC 61850-5 (2003)).

61850-7-4 (2003). As an example the LN circuit breaker, XCBR, contains one data class named *switch position* with a data name *pos* of the CDC Controllable Double Point - ”DPC”. The ”DPC” CDC is composed of a list of 20 data attributes. Each attribute has a name, type, functional constraint, trigger option, value/value range, and an indication of whether the attribute is mandatory or optional. An example of a specific data attribute for the ”DPC” CDC is the data attribute named ”stVal”, with an attribute type ”CODED ENUM” that can have a value of *intermediate-state*, *off*, *on* or *bad-state*.

3.5.1.1 Control class model

DATA that provides controllable data attributes, as in the case of ”DPC”, ”SPC”, among others, follow a specific standard control model. This control model consist of specific services and behaviors described with the use of state machines. These services are:

- *Select* / *Select with value*;
- *Cancel*;
- *Operate*;
- *Command termination*.

The available behavior models, or state machines, are four:

- Direct Operation with Normal Security;
- Select Before Operate (SBO) with Normal Security;
- Direct Operation with Enhanced Security;
- SBO with Enhanced Security.

In Direct Operation the *Operate* command is sent directly to the object under control. In SBO control, first a *Select* command must be sent that will put the machine in a privileged, exclusive, state (*Ready* state) and only then the *Operate* command can be issued. When in the *Ready* state, the *Cancel* command can be issued to resume normal operation (*Unselected* state). Enhanced Security differs from Normal Security in the *Command termination*.

3.6 ACSI and SCSM

The concept of the SCSM has been introduced to be independent from communication stacks including application protocols. One objective of the IEC 61850 series is the interoperability of devices. This requires that all communicating devices use the same communication stack. Therefore, the goal of this independence is not to have many mappings in parallel, but to be able to follow the state of the art in communication technology. This means that all the abstract information and service models defined in parts IEC 61850-7-4 (2003), IEC 61850-7-3 (2003) and IEC 61850-7-2 (2003) must be mapped to standardized communication systems as defined in parts IEC 61850-8-x. Already defined and in use by the standard, IEC 61850-8-1 (2004) part, defines the mapping to MMS, Transmission Control Protocol (TCP)/Internet Protocol (IP), and ISO/IEC 8802-3.

Referring at the ISO/Open Systems Interconnection (OSI) stack layers we can say that layers 1 and 2 are mapped to Ethernet while layers 3 and 4 into TCP/IP and layers 5 to 7 into MMS. MMS is written in Abstract Syntax Notation One (ASN.1) notation, refer to Dubuisson and Fouquart (2000) for literature on the subject, and it is mapped to TCP/IP by Request For Comments (RFC) 1006, Marshall T. Rose and Dwight E. Cass (1987). Time critical messages like Sampled Analog Values (SAV) and Generic Object Oriented Substation Events (GOOSE) are mapped directly to the Ethernet layer. GOOSE data is directly embedded into Ethernet data packets and works on publisher-subscriber mechanism on multicast or broadcast MAC addresses. Several other mechanisms are used in order to assure communication speed and reliability. SAV goes out of the scope of this work and is not examined here.

A brief but clear overview of the standard application is provided in Brand (2004). A slightly more detailed one can be consulted in Baigent et al. (2005). Readers further interested in the testing procedures can refer to Udren and Dolezilek (2006).

Chapter 4

Simulation Platforms and Simulation Models

4.1 Introduction

A model is a representation of an object, concept or system. This representation can be abstract or physical, static or dynamic. Generally, since most problems of interest in the real world are usually too complex, the model represents an abstract view where assumptions and simplifications are made of the complex reality. In this sense, simulation is nothing more than the manipulation of the designed model in order to mimic the real object, concept or system under study. A careful balance of the model complexity must be exercised. A too complex model might reduce the simulation performance to an impractical point. A too simple model might introduce errors soo big that the results are totally inaccurate.

The advent of the computer and its rapid growth and widespread use allowed the field of computer simulation to expand alongside. When traditional methods, like analytical solutions, are unable to provide results the computer simulation methods can still be employed. Computer simulation provides several benefits and advantages over pure analytical methods. It can be used in the optimization of systems, performance and/or reliability wise. It can also be used to verify correctness of designs before production. It can provide a virtual environment for training purposes. Different models for factories, communications and computer networks, integrated circuits, and all other sorts of systems have already been constructed. Some disadvantages like time consuming and expensive to build models, extensive training and experience required are also part of the simulation area.

In order to avoid the inherent disadvantages of using simulation, several platforms with different characteristics have been developed. Simulation platforms can be classified as either being discrete or continuous depending on the way the simulation objects are manipulated. In the discrete type of simulation the system operation is represented as a chronological sequence of events where each event occurs at an instant in time and produces a change of state in the system. This time progression can be event based (steps with variable length) or incremental (stepped, fixed length).

4.1.1 Simulation platform

The sole task of evaluation and decision for a specific simulation platform is a complete study in itself as also identified in Albrecht (2010). There are many choices each with it's own character-

istics. The project itself also has some very specific requirements and to the author knowledge there is currently no simulation models available for the Substation Automation System (SAS) field. The project also makes extensive use of communication networks and for that reason several network simulators were looked at in order to make a decision. The decision in this work ended up being on using OMNeT++ simulation framework. At the moment of this writing version 4.1 is the latest stable version and as such this was the used version. The next paragraph provides part of the reasoning that supported this decision.

The motivation of developing OMNeT++ was to produce a powerful open-source discrete event simulation tool that could be used by academic, educational and research-oriented commercial institutions for the simulation of computer networks and distributed or parallel systems (Varga (2010)). The fact that OMNeT++ follows a framework approach allows the independent development of OMNeT++ and the specific application areas frameworks. Already several university research groups and non-profit research institutions use OMNeT++ and companies like IBM, Intel, Cisco, Thales and Broadcom are also using OMNEST[1] successfully in commercial projects or for in-house research (Varga and Hornig (2008)). These were important factors for the decision but OMNeT++ also had important design requirements. Enabling large-scale simulation, simulation models need to be hierarchical, and built from reusable components as much as possible. The simulation software should be modular, customizable and should allow embedding simulations into larger applications. Data interfaces should be open: it should be possible to generate and process input and output files with commonly available software tools. Several simulation tools were also compared in Varga and Hornig (2008): NS-2, J-Sim, SSFNet, JiST and SWANS, OPNET and QualNet. NS-2 despite being one of the most widely used it does not provide some of the requirements for the project and/or introduce drawbacks. An important one being its focus only on network simulation and the fact the models are contained within the supporting infrastructure. J-Sim, SSFNet, JiST and SWANS projects have a reduced activity. OPNET is based on C and it's models are of fixed topology stored in a proprietary format. OMNeT++ is open-source modular and versatile.

4.2 OMNeT++ simulation platform

OMNeT++ is a component based, modular and open architecture for discrete event simulation. Because of its extensibility, on-line documentation and free academic license it is a popular

[1]OMNEST and OMNeT++ are largely identical, and simulation models written for one are guaranteed to compile and run with the other. Differences apply to licensing, packaging, and certain features only (Simulcraft, Inc. (2011)).

choice in academia. OMNeT++ itself does not simulate anything in particular but rather provide a common framework that can be used or extended by users to create any simulation model required. Because of this characteristic, domain specific functionality is developed and provided as independent projects. Currently there are already extensions for real-time simulation, network emulation, alternative programming languages (Java, C#), database integration, System C integration, and several other functions (OMNeT++ Community (2011)).

4.2.1 Overview

OMNeT++ is an event based simulation platform on which the events are modeled as messages. It's working principle consists on modules that communicate by passing messages between each other or themselves, triggering the events. The modules contain gates that are used for connection to other modules and for the transmission of messages. All sent messages go into a global queue and are time ordered. An entity called the scheduler is responsible to retrieve the messages from the queue and deliver them to their destination module. The lowest level modules are called simple modules and these can be grouped into compound modules to form an hierarchy. The hierarchy can have unlimited levels and the whole model, being itself also a compound model, is called network. A graphical representation of this hierarchical structure can be seen in Figure 4.1. In this example we can observe the Network module, simple modules connected by gates and a compound module made of different simple modules and a gate.

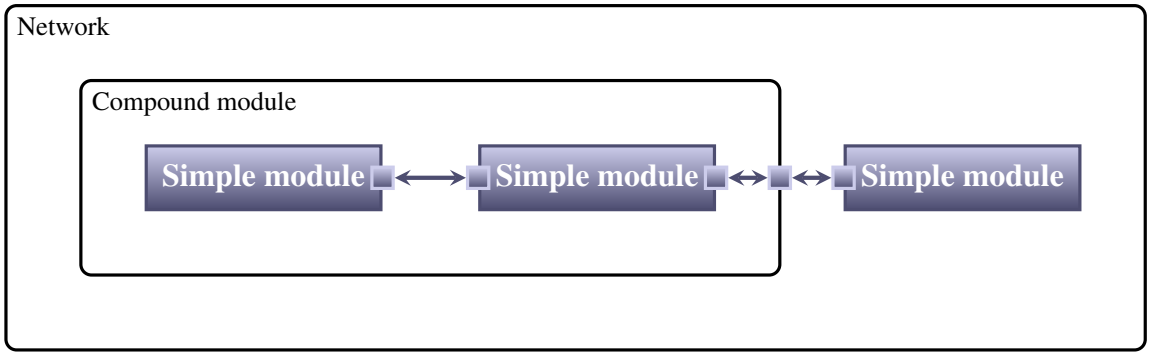

Figure 4.1
OMNeT++ simple and compound module types example. One Network composed by a simple and a compound module, itself composed of 2 simple modules. Compound modules agglomerate other modules, both simple and compound, while simple modules provide C++ implementations (omnetpp41).

All modules derive/subclass from the *cSimpleModule* class. This class provides several virtual implementations for specific modules to derive their own implementations. The logic behind it includes an *initialize* and a *finish* function that are called on simulation initialization

and finalization respectively. The *handleMessage* should contain all the internal logic required for the implementation by using the message sending/receiving functionality. Using this structure a timer can, for example, be easily implemented by programming the module to send a scheduled message to itself (self messaging concept) upon initialization. Listing 4.1 contains a small excerpt from the *cSimpleModule* class.

```
30 /**
31  * Base class for all simple module classes. cSimpleModule, although packed
32  * with simulation-related functionality, does not do anything useful by itself:
33  * one has to subclass from it and redefine one or more virtual member
34  * functions to make it do useful work. These functions are:
35  *
36  *    - void initialize()
37  *    - void handleMessage(cMessage *msg)
38  *    - void activity()
39  *    - void finish()
40  *
41  * initialize() is called after \opp created the module. Multi-stage
42  * initialization can be achieved by redefining the initialize(int stage)
43  * method instead, and also redefining the numInitStages() const method to
44  * return the required number of stages.
45  *
46  * One has to redefine handleMessage() to contain the internal logic of
47  * the module. handleMessage() is called by the simulation kernel when the
48  * module receives a message. (An alternative to handleMessage() is
49  * activity(), but activity() is not recommended for serious model development
50  * because of scalability and debugging issues. activity() also tends to lead
51  * to messy module implementations.)
52  *
53  * You can send() messages to other modules, or use scheduleAt()+cancelEvent()
54  * to implement delays, timers or timeouts. Messages sent or scheduled (but
55  * not cancelled) are delivered to modules via handleMessage(), or, when using
56  * activity(), via receive().
57  *
58  * The finish() functions are called when the simulation terminates
59  * successfully. Typical use of finish() is recording statistics collected
60  * during simulation.
61  *
62  * @ingroup SimCore
63  */
64 class SIM_API cSimpleModule : public cModule //implies noncopyable
65 {
66     friend class cModule;

92   protected:

115     virtual void handleMessage(cMessage *msg);
```

Listing 4.1

Partial C++ header file for cSimpleModule class. File taken from OMNeT++ version 4.1 source code.

4.2.2 Compiling and execution

An OMNeT++ model consists of several parts. Topology description files (NED files), message definition files (MSG files) and C++ code for simple modules implementation. In order to build a simulation program first MSG files must be translated into C++ code. The MSG files are written in a specific language and require parsing with a tool provided by OMNeT++. After this step is done, then all C++ code is compiled and linked together. The resulting executable dynamically loads the NED files and the simulation executes. NED files are written in a specific NED language. Inputs are provided by the use of command line arguments and also an INI file containing simulation parameters and program configuration options. The outputs can be analyzed using either command line tools or the provided Graphical User Interface (GUI) with features such as sequence chart diagrams.

Since NED files are loaded dynamically at run-time, after a simulation model has been compiled one can easily change them and execute different simulations, e.g. for testing different network topologies. On a properly conceived model this can be easily done because, besides being very descriptive and extensible, NED topology files are written in plain text. The output files are also easily exported since their contents follow a strict set of rules and they are also written in plain text.

Listing 4.2 presents an example NED file where one can observe the syntax for describing a network with sub-modules and connections. A concept not yet mentioned that can be observed on this example is that of a *channel*. The modules are not simply connected by gates but there is also a *channel* that represents the medium on which the messages travel between gates. In this example, the *channel* used accepts a datarate parameter since it attempts to represent a communications network.

```
1  //
2  // A network
3  //
4  network Network
5  {
6      submodules:
7          node1: Node;
8          node2: Node;
9          node3: Node;
10         ...
11     connections:
12         node1.port++ <--> {datarate=100Mbps;} <--> node2.port++;
13         node2.port++ <--> {datarate=100Mbps;} <--> node3.port++;
14         node3.port++ <--> {datarate=100Mbps;} <--> node1.port++;
15         ...
```

```
16 }
```

Listing 4.2
This NED code defines the architecture of a network type named *Network* with nodes of type *Node* and it's connections (Varga (2010)).

4.2.3 Architecture

The platform is composed of several different libraries. *Sim* contains the simulation kernel and class library. *Envir* contains the code common to the different interfaces. Also in this library the main application entry point can be found. The main purpose of *Envir* library is to interface with *Sim* and the executing model while hiding the interface implementation details. This way one can replace the interface, easily without having to touch other libraries. *Cmdenv* and *Tkenv* are specific user interfaces implementations provided by default with OMNeT++. *Cmdenv* is a command line interface that executes in the console/shell and *Tkenv* is a GUI implemented in TCL/TK. Tool Command Language (TCL) is a programming language and Tk is a graphical user interface toolkit used by TCL. The Model Component Library and the Executing Model are not really libraries in itself but rather abstractions from the general principles of OMNeT++ platform. OMNeT++ does not provide any Model Component by default. The user must built it's own components by deriving from the *cSimpleModules* or using already built frameworks like *INET*. The Executing Model is built during execution by reading the NED topology files and loading the necessary models from the library. This general architecture overview of OMNeT++ can be observed on the graphical representation in figure 4.2.

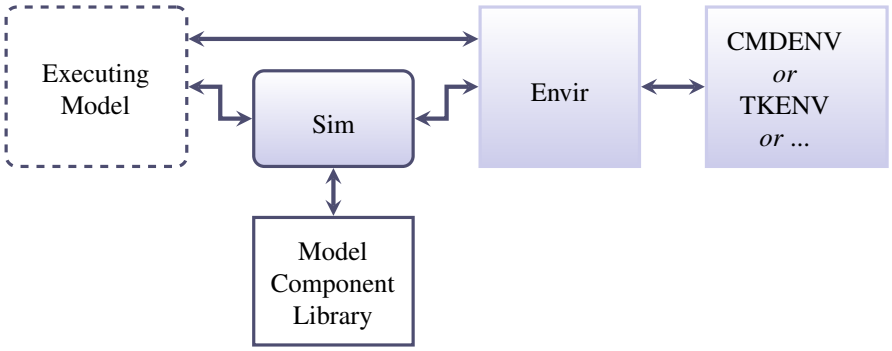

Figure 4.2
OMNeT++ architecture (Varga (2010)).

OMNeT++ also depends on external libraries like pthread, libxml2, zlib, iconv and tcl/tk and other more peripheral/utilitarian libraries like *common*, *layout*, etc., not represented here.

4.2.4 Extensibility

No single platform can provide the needs of all user simulations. OMNeT++ accepts this and is built with extensibility in mind by using Object-Oriented Design (OOD) principles. Any library and/or part of a library is easily replaced and/or extended provided one follows the existing Application Programming Interface (API) interface. Commonly one could wish to extend the output classes, in order to use a specific output format or to have a direct connection with some output processing application. Also extending and/or replacing the user interfaces. And complete integration with another application by embedding OMNeT++ simulation into it.

OMNeT++ manual provides some examples and starting points/guidelines in order to extended and embed the simulation framework. On listing 4.3 one can see a possible example implementation of a user interface by deriving from *EnvirBase*.

```
1 #include "envirbase.h"
2
3 class FooEnv : public EnvirBase
4 {
5     ...
6 };
7
8 Register_OmnetApp("FooEnv", FooEnv, 30, "an experimental user interface");
```

Listing 4.3
Example user interface implementation. Example taken from version 4.1 of OMNeT++ on-line manual.

Chapter 5

Development

5.1 Overview

In order to better define the context of this work lets first start by looking at the Energy Power System (EPS). After, the substation and its automation systems will be introduced with the currently available tools, its current limitations and finally the project contributions. We start by making simple assumptions and providing practical examples and slowly build the complexity towards the project goals and objectives.

5.1.1 The Energy Power System

The EPS is composed of several substations. These substations can be geographically spread and they are interconnected to form e.g. the EPS transmission system. It's main purpose is the bulk transfer of electrical energy, from generating power plants to substations located near population centers. This means that these connections form a network but refer only to electrical conductors. However, there is also a requirement to control and manage this network and consequently the substations. This happens, for instance, when a failure occurs in one of these substations. In order to prevent such a failure to propagate through the entire network (grid) and cause a major blackout (cascading failure), mechanisms to prevent it must be in-place. This is part of the functions of the control and management system. As an example, this system can enable parts of the network to be disconnected to prevent further fault propagation. Later the failing equipment can be fixed and/or replaced and the concerning network part reconnected again. This restores the energy back to the affected areas and prevents other areas from even noticing the problems. A simplistic, but appropriate, view of the control and management system, is that of a main Control Center (CC) connected via a communications network to one or more substations, like figure 5.1 shows. From the CC, the transmission system grid can be monitored, and controlled if necessary, by an operator (user). This user can be a human or an automated system. Currently most protection schemes involve an automation system and the human operators are relegated to monitoring and management functions. Usually the interface between the user and the system involves a single or a network of computers running a Graphical User Interface (GUI). It is commonly referred to simply as a Human Machine Interface (HMI). The HMI GUI must present to the user, in an easy, clear and fast way, all the information he needs to perform his duties effectively and efficiently. This usually involves some graphical

representation of the network substations (nodes) and its connections. It is also common to contain other relevant information like current system states and values, e.g, connected/disconnected, nodes voltage and/or current, etc. It can be a complete or partial representation of the network depending on the user needs. This global picture forms what is commonly known as a Supervisory Control And Data Acquisition (SCADA) system. This acronym can, and in fact is, also used in other systems besides the Energy Power System (EPS), like industrial processes. Figure 5.1 contains a representation of a possible EPS with both the electrical connections for power transmission and the communications network for data acquisition and control.

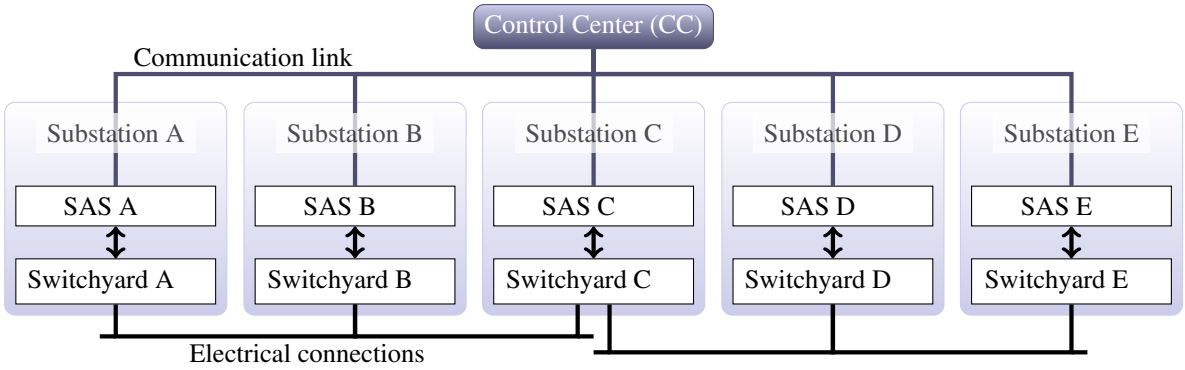

Figure 5.1
Energy Power System diagram with the electrical (bottom) and the communication (top) networks. The Substation Automation Systems (SASs) control and maintain the electrical connections in the switch-yard. Control commands can also be sent from the Control Center (CC) or other SASs for coordinated control.

5.1.2 The substation

The substation is composed of several different equipment. These equipment is divided into two categories commonly named *primary equipment* and *secondary equipment*. From the previous section one can also identify two different type of connections, to and from the substation: the electrical connections and the communication link that is part of the control and management system, figure 5.1.

The primary equipment category groups all the equipment concerned with the electrical conduction and transformation. From figure 5.1 this means all the equipment in the switch-yards. From the previous example this refers to the switch that connects, or disconnects, the substation from the rest of the grid. This includes also other type of equipment besides the switches but, commonly, the equipment and the space where it is located are named switchgear and switch-yard, respectively. It is common to find sources that refer to both only as switch-yard.

The secondary equipment refers to all other types of auxiliary equipment necessary for the operation of the substation and that are not connected directly to the electrical connection. From figure 5.1 this means all the equipment that is part of the SASs. In the example this refers to the device that is connected to the Control Center (CC) via a communication link and to the switch via some other method. When the appropriate command is sent from the CC the device operates the switch. These devices that receive and interpret the commands and interact with the primary equipment are named Intelligent Electronic Devices (IEDs).

Inside the substation the IEDs are connected to a local communication network and to the switchgear (figure 5.3). It is common to connect IEDs to the switchgear with a serial link. Sometimes an optical link is used to avoid signal interferences due to the high voltages present in the switch-yard. Usually the local network interacts with the main communication network, from the control and management system, via a gateway device. It is also frequent to have different communication protocols being used by the local and global networks, meaning that, the gateway device must understand all communication protocols in use and act appropriately. Sometimes inside the substation there is also a local control room. This is usually referred to simply as a Human Machine Interface (HMI). Just like the CC control room this HMI can support a GUI for local control and management. Commands issued from the CC are usually referred to as remote (control) from the substation point of view. The commands issued locally (local control) take precedence over remote. The reasons for this are mainly of security and safety nature. If a human needs to operate in the switch-yard, a change of state issued from the main CC, unaware of such human presence, would cause serious danger. For this reason, functions for interlocking are used extensively inside the substation. Continuing the previous example, for human safety the switch could be connected to an interlocking device. This interlocking device would stop any remote control if certain conditions are met. This could include the position of another switch. Before entering the switch-yard, any human operator would flip this switch and prevent the remote operation of the switch-yard, safeguarding his own life.

5.1.3 The Substation Automation System

The functions of a Substation Automation System (SAS) refer to tasks, which have to be performed in the substation. These are functions to control, monitor and protect the equipment of the substation and its feeders. In addition, there exists functions, which are needed to maintain the SAS, i.e. for system configuration, communication management or software management (IEC 61850-5 (2003)).

The previous sections mention two important functions of the substation: switching and interlocking. These are however, only two of the control functions for the substation. There is

also the need to take into account different types of substations that require different functions from each other. A substation connecting a generating plant to the grid is certainly different from one connecting hundreds of consumers to the grid. Substations can have different topologies, equipment and purposes.

Humans are usually slow and error prone when compared to a machine. For this reason most of the functions inside the substation are automated. The Substation Automation System (SAS) is the name of the system containing all of the devices involved in the automatic control of the substation. This also accounts for commands generated by a human operator that are then processed by the SAS.

5.1.4 The Substation Automation Systems standard

The IEC 61850 standard describes a data model of the substation while taking all of its requirements into account. Its data model provides all the information necessary for control and protection functions but also the IEDs and the switchgear configuration. The standard also promotes a standardized engineering process and includes a comprehensive protocol conformance testing procedures. A description language, Substation Configuration Language (SCL), written in Extensible Markup Language (XML) was also developed to promote the exchange of information between devices but also with other tools. Using an Object-Oriented (OO) paradigm, taking into account future extensibility, the model data and services, Abstract Communication Service Interface (ACSI), are separated from the communication, Specific Communication Service Mapping (SCSM). This allows both to develop and evolve, keeping up with the latest technological developments, at their own pace without conflicts. The substation functions are decomposed into smaller components, named Logical Nodes (LNs), and the standard allows the distribution of these among several devices (IEDs).

These IEC 61850 features are also summarized in Brand et al. (2004). They not only cover the communication protocols but also the entire substation engineering process.

5.1.5 KEMA testing and certification tools

KEMA is one of the companies currently involved in the testing and certification of the IEDs regarding IEC 61850 standard. The target of such tests consider the internal object modeling, data and services, and online configuration. Several third-party testing laboratories are also certified and regulated fr this standard, but KEMA laboratory has the highest possible classification level. Inside KEMA a tool named UniCA Client Simulator, referred to as the client or client simulator from now on, has been developed and is used extensively for such purpose. The client, usually

running on a computer, is connected to a single IED. Several test cases, as specified in the standard, are then executed. These involve the interaction between the client and the IED where for certain actions from the client certain reactions from the IED are expected. The client is then able to execute these actions, that are written using a scripting language, and to analyse the reactions from the IED. The actions involve the communication mechanism and the exchange of messages from the standard domain. During testing a network analyzer is also used to monitor and log all messages in the network. At the end, if all conditions are met, a certificate can be issued to the specific IED stating it's conformance with the IEC 61850 standard. This complex and thorough testing process also include mechanisms to report and to correct possible flaws and/or faults detected in the standard.

5.1.6 The limitations

A generic testing plan includes four categories: conformance testing of the devices (IEDs), the distributed functionality testing, the interoperability testing and the global system performance (Haffar et al. (2010)). Currently the standard testing procedures ensures the conformance of IEDs but it still cannot guarantee its *distributed functionality*, *interoperability* and global *system performance*. Devices, e.g. from different vendors, may conform to the standard but still be unable to inter-operate together in a seamless and fault free way. The standard does not specify any functional logic, i.e. what should be, for instance, the algorithm and/or logic for interlocking. This fact and the lack of logic modeling contributes to the difficulty of ensuring interoperability and maybe interchangeability of devices (IEDs). Different vendors may implement different strategies and philosophies and there is no standardized way to describe the logic used in specific IEDs. Some efforts are currently being done in order to include this in future editions of the standard (Moreno (2010)). The limitation resultant from the conformance only testing is also highlighted in Haffar et al. (2010). The remaining tests, distributed functionality, interoperability and system performance can all be viewed as SAS functionality and performance testing.

5.1.7 The contribution

The current project aims, on a first stage, at developing a tool capable of Substation Automation Systems (SASs) functionality and interoperability testing and later, on a second stage, it will be extend into performance testing also. The SAS functions that this project test system will cover initially are the switching and interlocking. These two functions are widely used in all

substation projects and providing an easy to use and automated test system will contribute to significant savings in both time and money.

5.2 Test system definition

Having defined our goal to test the Substation Automation System (SAS) and its functionality we shall begin the problem definition by looking at it as a black box, figure 5.2. On top, the SAS communicates with the main Control Center (CC) by a communication network. On the bottom it is connected to the switchgear in the switch-yard by using several, possibly optical, serial links. In order to completely test it, one must be able to monitor and control both the CC and the switchgear. The CC might be interested in a specific state of a specific switchgear. By controlling both, the test system is able to change this switchgear state and then to expect the correct state to be communicated back to the CC. Now to further develop this we must start looking inside the SAS, see figure 5.3. The SAS itself is composed of several different components. The typical SAS will contain one or more IEDs, a Gateway and a HMI all connected together in a local network.

Gateway: The Gateway is mainly a device that interconnects the communication networks from the main CC with the local network. Commonly these networks are of different nature and also use different protocols. In this sense the Gateway acts as a translator and also a barrier. In certain situations the Gateway can itself be also an IED.

IED: The IEDs are mainly the devices that operate the switchgear. Therefore they must be connected to the local network and also to the switchgear. This is not always the case, if e.g. the function of a specific IED is to record and store in a database the transient voltage values of a specific switchgear. In the case that there is already another IED connected and serving this information from the switchgear (server role). Then the IED only requires a connection to the local network and to the database. The required information is then acquired from the server IED in the local network (client role). Client-server relationship between both IEDs, paradigm used in the IEC 61850 standard.

HMI: Just like the control room from the CC certain substations also provide a local human access for interaction with the SAS and the equipment. This is mainly just referred to as the HMI but usually it consists of a room with a computer running a GUI.

5.2.1 Test system requirements

Following the work proposed in Schimmel (2008) the basic requirements for such a testing system are summarized as effectiveness and efficiency.

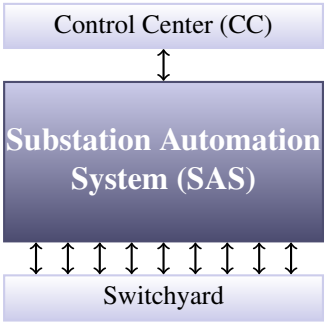

Figure 5.2
Substation Automation System (SAS) as a black box. On top it communicates with the Control Center (CC) while on the bottom it's hardwired to the switchyard (Schimmel (2008)).

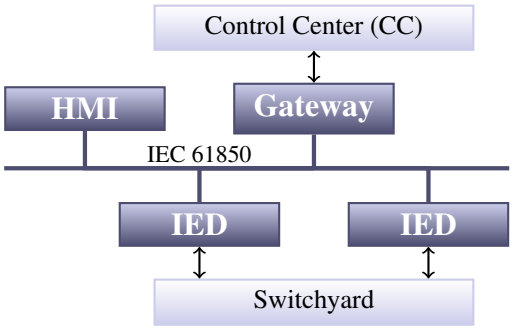

Figure 5.3
Substation Automation System (SAS) inside view. The different components of the Substation Automation System (SAS) (Schimmel (2008)).

Effectiveness refers to the market value and the contribution of the test system to the field. An automated test system capable of moving time spent on Site Acceptance Test (SAT) to Factory Acceptance Test (FAT) provides significant savings to all substation projects. Automated test procedures improve the time spent on testing but also reduce the probability to introduce new errors. This is also mentioned in Haffar et al. (2010) as a key demand from end users (utilities). Testing of switching and interlocking, being two widely used functions, allows such an automated system to capitalize the needed improvements on time spent testing.

In order to be efficient the test system requires to be flexible and automated. Automation must enable fast test execution, reproducible test results, archiving and analysis of test results. Flexibility must enable the application of the test system to many different systems while providing an easy configuration. Requirements such as no, or minimum, human intervention and no need for highly experienced test engineers contribute to both flexibility and automation. Other requirements include the ability to simulate missing devices, analysis of 61850 messages and a time stamp accuracy of at least 1 millisecond. According to the standard definition every message contains a maximum response time or Time To Live (TTL). Certain messages, according to importance, have only a few milliseconds TTL hence the time stamp accuracy of 1 millisecond. A slower performing test system could not guarantee such response times and properly test the system. The ability to simulate missing devices makes the system scalable allowing the testing of small parts before the complete SAS test. The purposed solution in Schimmel (2008) for flexibility is to build the test scenarios as scripts and having a pre-built collection of templates. The executable tests are then generated as necessary from the templates. Because they are meant

to be written using a high level scripting language this allows the easy adjustments for specific details of certain SAS. The easy configuration is achieved by using the configuration language provided by IEC 61850 standard, SCL. The necessary file would be the Substation Configuration Description (SCD) file that contains the complete description of the substation and its devices already configured. This means that in this XML file the complete substation architecture with the switchgear and it's connections are described. The IEDs, with brand and model, and its complete description according to Logical Devices (LDs) and Logical Nodes (LNs). It is also described the relation between each switchgear and the devices LNs it interacts with.

5.2.2 Test system components

From these requirements the complete test system can then be decomposed into several components. This decomposition can be observed graphically in figure 5.4.

Test master - Test scripts: The Test master is the core of the testing system. Its main purpose is to read the different test scripts and execute them. For that he will convert the specific script actions into commands to the different components. Feedback from the components can also be received and appropriate actions executed. An example application would be to test the switching operation of a specific switchgear from the local Human Machine Interface (HMI). The script template would state the generic actions to take, i.e. send operation command to the IED that controls the switchgear and expect the switchgear state to change into a specific value within a specific amount of time. If this final state has the expected value and this occurs within the allowed time, then report the test case as success; otherwise as failure and state the reasons. From this generic template the test system script engine must transform it into concrete test cases for each of the switchgear present in the specific substation under test. This means resolving and referring specific switchgear and also the IEDs connected and related to the switching functionality of them. Then communicate to the different components such actions, i.e. for the current example telling the HMI simulator component to the send the command to the controlling IED. After that expect the feedback from the I/O system component reporting the switch state changed within the time limit.

I/O System: Real IEDs connected to the switchgear must also be tested. In order to be able to test them the switchgear must be emulated. There are many different types of connections most using serial communication, e.g, a connector with 100 V and 5 A or even optical links. Several manufacturers already provide such devices that can be connected to IEDs and that emulate the switchgear. This component is meant to represent such devices.

I/O driver: Since several different types of I/O Systems exist, the test system must be able to interface with them. For that a driver implementation allows the test system to load the necessary code for each of the different I/O Systems, hence the name of the component.

IED simulator: Several conditions support the need for a scalable test system, namely, testing smaller parts of a complex substation first, or even trying to decide which IEDs to acquire before the investment is made. For this to be possible the missing IEDs must be simulated. This component responsibility is to simulate the behavior of a specific IED. Since each device is different, different vendors and even different models behave differently, this component must also take that aspect into consideration. Having an IED library from different vendors supports this requirement.

HMI simulator: Part of the functionality of the SAS is to provide information and interact with the local HMI (control room). Therefore this must also be tested. This is depicted here with the HMI simulator component.

CC simulator: Like the HMI simulator, the CC simulator purpose is to simulate the functionality used to/from the CC to the SAS. In reality, regarding IEC 61850 standard, the client-server model used doesn't make any distinction from the server to the client point of view. This means the HMI and the CC (even other IEDs) access a server (an IED providing information) the same way. The only difference here is that the HMI has direct access to the local network. The CC simulator must, depending also on substation, access the Gateway device first. In the case the communication from CC to SAS is not IEC 61850 based and the intention is also to test the Gateway then the CC simulator must also be able to understand and use the other protocols in place, as it was mentioned earlier.

Analyzer: As with any true test or experiment, results must be recorded. For this reason the Analyzer component exists. It's main job is to monitor the network and record everything, or everything relevant, that is being transmitted in the local network. These results can be stored and analyzed later.

5.3 Test system prototype

OMNeT++ was the chosen simulation platform for the test system prototype. Based on the previous test system definition, figure 5.4, the following components have been implemented and included in this prototype version.

Implemented Components:

- *Test master – Test scripts*: The test scripts feature from this component will not be implemented but certain small developments must be done in order to execute properly some initial tests.

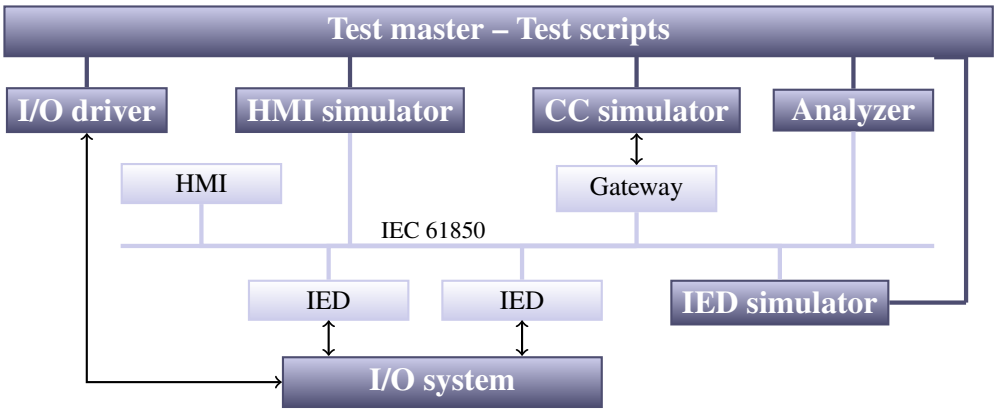

Figure 5.4
Test system definition components (Schimmel (2008)).

- *IED simulator*: The IED simulator will be implemented by integrating the Dynamic Link Library (DLL) interface that contains the core functionality from KEMA tool, UniCA Multi-IED Simulator.
- *I/O driver*: This component is the interface between the test system and the I/O system. The I/O system will not be included in this version of the prototype but, despite the name, it is also implicit in this component the simulation model for the primary equipment. This simulation model was implemented under the name of *process model*.

The remaining components will not be included in this first stage prototype version but are planned for later development.

Future versions components:

- *I/O system*: This component requires hardware development or integration with hardware currently available in the market. Because of the time necessary to development such a system the obvious choice goes to the integration of third party hardware. Several devices already exist in the market and a study to decide which ones to support is required. This choice will affect the I/O driver component because specific drivers must be written to become part of the I/O driver library.
- *Analyzer*: The job of the analyzer is to record all communications in the local network. This initial version of the prototype will not devote time developing or integrating this component with other tools. Instead, one of the already available tools in the market was used, KEMA UniCA Analyzer, by executing it as an independent process.
- *CC simulator*: This initial version of the prototype will not develop this component and KEMA UniCA Client Simulator will be used to replace it in the initial tests. This tool will be executed as an independent process and commands emulating the scripts will be issued from it.

- *HMI simulator*: This initial version of the prototype will not develop this component and KEMA UniCA Client Simulator will be used to replace it in the initial tests. This tool will be executed as an independent process and commands emulating the scripts will be issued from it.

The following section describes with more detail the development and implementation of the different components.

Test master – Test scripts: The Test master should be able to read the test scripts and execute actions accordingly. It also needs access to the remaining components of the simulation environment in order to interact with them. In OMNeT++ the scheduler already performs such a similar central role. The implementation extended the scheduler in order to accommodate for the new requirements. The scripting functionality, reading and executing scripts, was not implemented. This requires a complete study on methods and languages and for that reason it was not included in the initial prototype.

Scheduler: In order to extend the scheduler to accommodate the test system requirements some changes had to be made. The requirements include the communication between external devices. OMNeT++ simulation works internally by the exchange of messages between modules. When interfacing with external devices, like real network interface cards, the scheduler is responsible to monitor such devices. When something relevant to the simulation happens in the devices the scheduler must generate the appropriate simulation model messages and deliver them to the relevant modules. This means the scheduler is tightly connected to the I/O driver and I/O system of the test system. Some termination conditions, that will be explained in more detail later, had to be implemented in the scheduler for test termination.

Configuration: In order to keep the system configuration process simple, according to the requirements, the initial input to the test system is the SCD file. OMNeT++ requires different files formats for configuration (NED, MRT, INI). The different files must then be generated dynamically from the initial SCD file. This required the development of a set of functions, grouped into a library called *scl2ned*. Figure 5.5 shows a graphical representation of this resulting process flow.

I/O driver: The I/O driver as described, is mainly a driver interface to other external devices that emulate the switchgear (I/O system). In reality several other implicit requirements are necessary and part of the I/O driver. The switchgear must be simulated and that simulation model must interact with different types of I/O systems but also with the IEDs simulation model. By using an abstract Application Programming Interface (API) for DLL development enables the loading of different device drivers at run-time. This also detaches the development of such drivers from the main test system allowing the creation

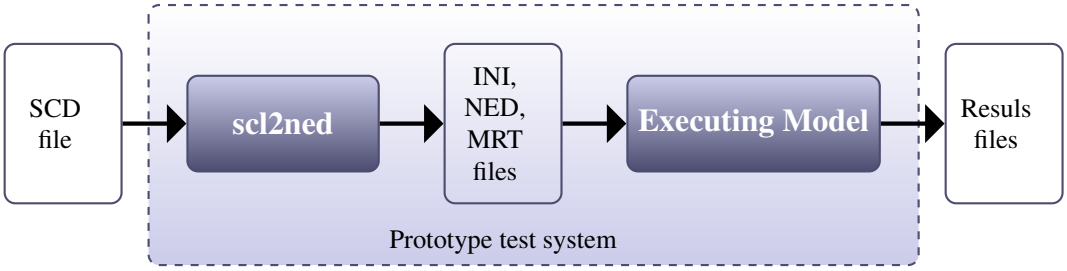

Figure 5.5
Prototype process flowchart. A SCD file is ”fed” into the prototype that will generate the necessary files to execute the simulation.

of an independent I/O driver library. The simulation model for the switchgear is referred to as *process model*.

Process model: The process model must simulate the different switchgear and its connections to the IEDs. These IEDs can be both simulated and real devices. The simulated IEDs are part of the *logic model* simulation model. In this case function calls from the process model to the logic model were used. In the case of real devices a mechanism to interface with external devices that emulate/simulate the switchgear is necessary. These devices will then be physically connected to the real IEDs just like in the real substation environment. Such devices are already available by different vendors, such as OMICRON®. Having an independent I/O driver library mechanism accounts for modularity, encapsulation and flexibility.

The process model classes were roughly based on the class models and hierarchy from the SCL structure. Figure 5.6 shows a short graphical representation (Unified Modeling Language (UML) class diagram) of the classes hierarchy used in the process part. The main functionality implemented in the *Switch* class enables the opening and closing of the switches. Two types of switches exist, disconnectors (DIS) and circuit-breakers (CBR), but in the current model both contain the same functionality. The class division of *ConductingEquipment* and *Equipment* relates to equipment with one terminal connection only, such as voltage measurement equipment, and equipment with two terminals (conducting equipment such as current measurement).

IED simulator: The IED simulation model is part of the *logic model*. The logic model must simulate different IEDs models and behaviors but also the SAS local network. By using also a driver mechanism with an abstract API for DLL run-time loading it is possible to develop different drivers, independently, each with its own characteristics. This IED library allows different development paths for different IEDs with different behaviors.

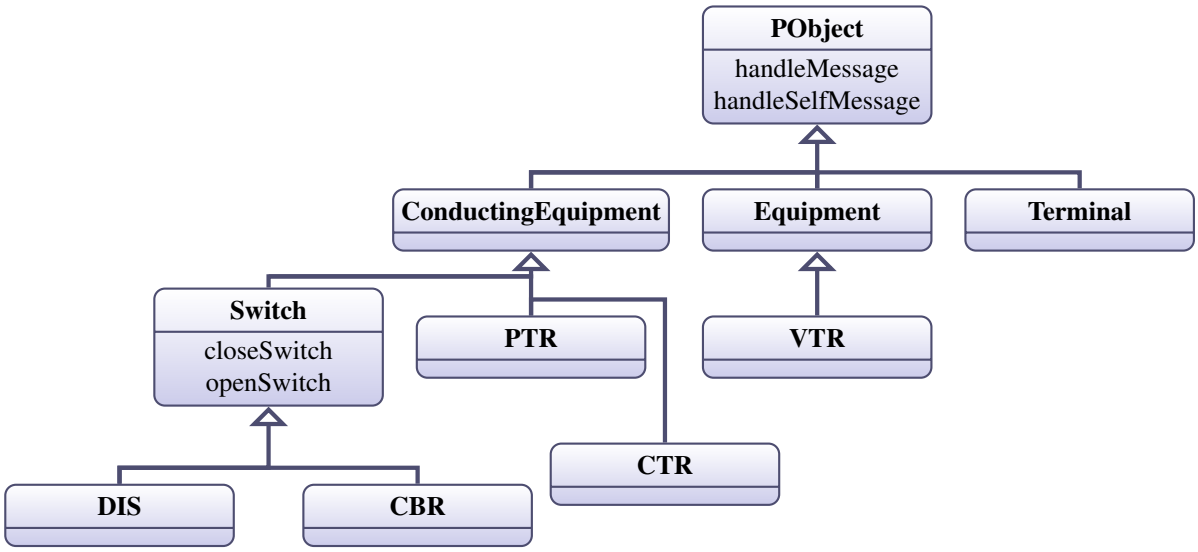

Figure 5.6
EPS process class structure. Simplified UML diagram.

Logic model: The logic model must simulate the secondary equipment and the communications network that connect them. These consist of multiple IEDs and a Ethernet network. The IEDs can be both real and simulated devices. The real devices must be connected to the computer system network executing the test system. This means that a mechanism to interface the simulation model with real networks must be provided. Initial development accounted for the complete development in OMNeT++ for such a simulation model allowing the interface with real networks. A framework for OMNeT++ containing such models for Ethernet networks, among others, already exists and some initial interface with real devices is already included. This library is called INET and the real networks interface is done using the *WinPCap* library and extending the scheduler implementation. For the network hubs and switches the default INET module *EtherSwitch* was used. This is enough for this initial prototype version but later a more detailed study must be done. For the IEDs network, components from INET regarding Ethernet networks were used and contained in a OMNeT++ compound module. This compound module was named *NetworkLayer* and is part of the *IED* compound module, figure 5.8.

The IEC 61850 standard does not provide any logic modeling requirements. Being so, the simulation model must account for different IED models. For this a similar interface mechanism, as in the process model, using a standard API driver mechanism with dynamic loadable DLLs was implemented. This allows the independent development of different drivers, each to emulate a specific device model behavior. The driver mecha-

nism is based upon the OMNeT++ simple module structure of *initialize*, *handleMessage* and *finish* functions. This driver system was built inside the *IEC61850* simple module implementation. Upon initialization the *IEC61850* loads the specified driver DLL.

In order to speed up development and quickly provide a simulated device for testing, a driver based on KEMA UniCA Multi-IED Simulator was implemented. This driver was named *kemaieddrv* and it will be further detailed below in section 5.4.4. This enabled to skip the complete communication stack implementation in OMNeT++ and as such the use of INET was not further explored. This is only necessary for a complete simulation environment tool and since this is currently not the goal of this project it was not pursued. Since this is also a useful feature the implementation was still done with INET functionality included for a future development. The tool provided by KEMA already contains the protocol implementation and the interface with a real network interface. A special driver interfaces with the logic model that provides the missing functionality. Listing 5.1 contains part of the developed driver API for the logic model. The functions *read_dataset* and *write_dataset* provide internal access to the IEC 61850 network stack.

```
44 // DLL entry function (called on load, unload, ...)
45 BOOL APIENTRY DllMain( HANDLE module, DWORD reason, LPVOID reserved );
46
47 // Exported functions and function pointers
48 typedef void (*FP_INITIALIZE)( device_info_t dev );
49 IED_DRV_API void initialize( device_info_t dev );
50 typedef void (*FP_HANDLEMESSAGE)( device_info_t dev, msg_info_t msg );
51 IED_DRV_API void handleMessage( device_info_t dev, msg_info_t msg );
52 typedef void (*FP_FINALIZE)( device_info_t dev );
53 IED_DRV_API void finalize( device_info_t dev );
54 typedef void (*FP_READ_DATASET)( device_info_t dev, node_info_t node );
55 IED_DRV_API void read_dataset( device_info_t dev, node_info_t node );
56 typedef void (*FP_WRITE_DATASET)( device_info_t dev, node_info_t node );
57 IED_DRV_API void write_dataset( device_info_t dev, node_info_t node );
58
59 #endif //_EPS_DRIVERAPI_
```

Listing 5.1
When a driver name is specified the *IEC61850* module will load the driver DLL and interface with it using this API.

5.4 Architecture and libraries overview

The complete architecture with all the libraries and relations between them can be seen in figure 5.7. Libraries with a blue background were developed for this project. *GUI* library was developed in C# since this allowed for a faster development of a functional GUI. Libraries

scl2ned, *EPSEnvWrapper* and *WinPCapWrapper* were developed in C++/CLI taking advantage of the vast .NET® framework for faster deployment. The wrapper libraries exists only to provide access to unmanaged code (C++) from within the GUI .NET® framework managed code (C++/CLI and C#). The remaining libraries were all developed in C++ (unmanaged). The EPS and the INET library contain the simulation models. EPS is the name given to the OMNeT++ library that contain the *logical model* and the *process model*. The executing model are instances of objects based on the building-blocks (modules) provided by the simulation models, EPS and INET. This obviously change from substation to substation depending on their configuration and therefore the generated NED files.

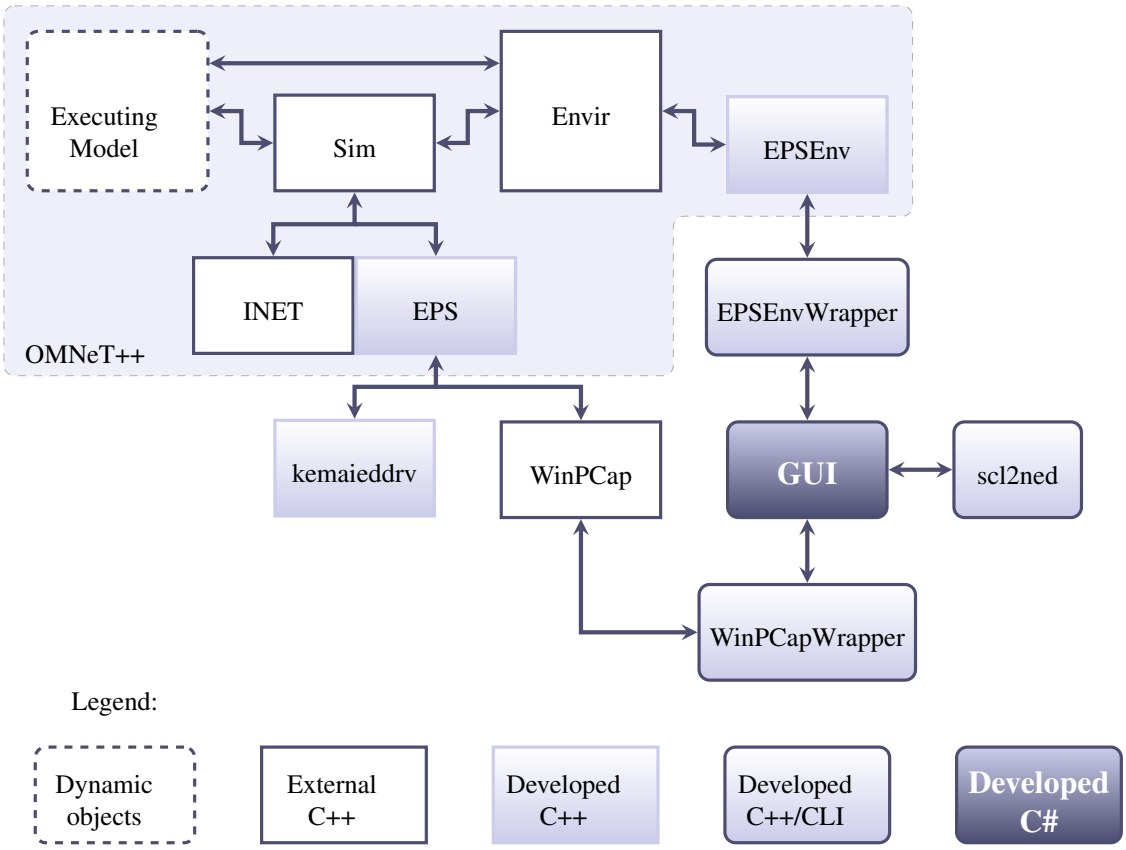

Figure 5.7
Test system prototype architecture. All developed libraries presented with a blue background. The executing model is composed of dynamic objects created at runtime from the NED files.

5.4.1 INET simulation model

In order to simulate the communications network and to communicate with external devices the INET framework version 20110225 was used. Quoting the INET website for it's description:

> "The INET Framework is an open-source communication networks simulation package for the OMNeT++ simulation environment. The INET Framework contains models for several wired and wireless networking protocols, including UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, 802.11, MPLS, OSPF, and many others."

Containing most of the communications network models necessary it is a perfect choice for initial development and prototyping. The provided scheduler example for real time interfacing was expanded to accommodate the project needs. This interface is done in Windows by using the *WinPCap* library. Version 4.1.2 of this library was used. Some of the modifications required include changes in the *WinPCap* and *WinSock2* logic in the *scheduler*.

This implementation was later dropped in order to use the core functionality from KEMA UniCA Multi-IED Simulator. This allowed a faster development time since this already includes the communication stack (IEC 61850, MMS, RFC1006, etc) and the interface with real network devices.

5.4.2 EPS simulation model

In order to simulate the substation a new OMNeT++ framework (simulation model) named *EPS* was created. This simulation model is in fact two different models, one for the primary equipment and another for the secondary equipment (IEDs) and its communication network. The *process model* and the *logical model* simulation models. The process refers to the substation primary equipment, the switchgear, the electrical terminal nodes, the bays, the voltage levels, etc. The logical refers to the substation secondary equipment, the IEDs, and it's logical components according to the standard, the Logical Devices, Nodes and Data Attributes.

5.4.2.1 Process model

5.4.2.1.1 Compound Modules The scl2ned library builds the compound modules according to the modularization options provided by the user. By default no modularization is included and all the primary equipment, simple modules, are connected directly to each other. The user has the ability to choose the bay, voltage and substation or any combination of these as modularization levels. This groups all primary equipment within these levels inside the same compound module.

5.4.2.1.2 Simple Modules The following simple modules were developed for the process model.

CBR and DIS (Switch) Both the CBR and DIS simple modules are subclasses of Switch and in essence they have the same functionality. In real life the difference between both is that the CBR (circuit breakers) can also operate under fault conditions in order to open short-circuits. For the current simulator operation they both operate the same way but both simple modules were developed in order to account for future needs and to keep the structure similar to real-life and the way the standard describes it. Table 5.1 and 5.2 contain the module events and main functionality description.

Table 5.1
Switch (CBR and DIS) simple module events.

Event	Origin	Description
Open Switch	XSWI LN	Operate the switch to the open position.
Close Switch	XSWI LN	Operate the switch to the closed position.
Timeout	Self	Self-message event to model the switch failure operation.
Operate	Self	Self/message event to model the switch operation.

Table 5.2
Switch (CBR and DIS) simple module functionality.

Action/Event	Description
Operation	When the events to operate the switch, closing and opening, are received the timer events Timeout and Operate are setup (scheduled) and the switch state is changed to indicate the in-transit operation. The predefined time values for the timeout and the operate are 10 milliseconds and 30 seconds and guarantee the normal switch operation, i.e. the timeout is bigger than the operate. In order to simulate specific equipment with operation times different than the defaults these need to be changed, this is done using the NED parameters feature that can be used to override the predefined values. They can also be easily changed to non-deterministic in order to simulate switch failures probabilities. Currently they also assume that the opening and closing operations of the switch take the same amount of time, this can be easily extended in the future.
Operate	When the operate event is received the switch state is changed to reflect the new stable state, open or closed, and the timeout event is removed from the scheduler.
Timeout	When the timeout event is received the switch operation did not occurred as expected, within the time limit, and the switch is put into the failure state. The operate event is removed from the scheduler.

Others The remaining simple modules, PTR, CTR and VTR, do not contain any functionality yet as they are not relevant for the switching and interlocking. They were implemented just in order to keep the real life and the standard structure. Also the scl2ned library will use the class names of the standard directly from the SCL file into the NED topology file so if those classes already exist as simple modules, even if empty, one does not need any logic mechanism to convert the names. The ConductingEquipment class specify the equipment with only two terminal connections. The Equipment class specify the equipment with only one terminal connection. The Terminal class allows multiple equipment to be connected together.

5.4.2.2 Logical model

The logical model was built incorporating the already existing INET modules for communication. A compound module named *IEDNetwork* was created based on the INET modules for networking. This module is then contained in the compound module named *IED* that also contains an *IEC61850* simple module. This implementation is partly exposed in listing 5.2 and can be graphically observed in figure 5.8. The most relevant detail is inside the *IEC61850* simple module. This module is provided with a specific driver DLL and upon initialization it loads this module to simulate a specific vendor and model IED behavior. The INET module chosen to be used for the network switches was the *EtherSwitch*. This decision will require a more detailed examination later but for a working prototype/example is enough.

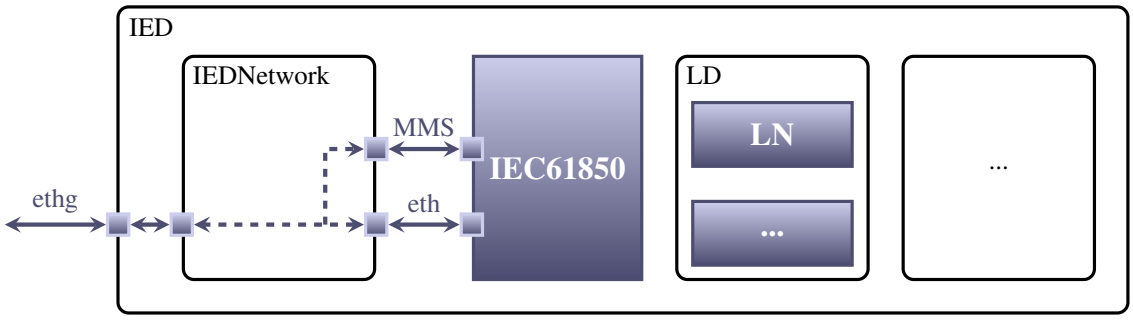

Figure 5.8
IED NED module developed for *EPS* simulation model.

```
13 module IED
14 {
15   parameters:

29     submodules:
30     IEC61850: IEC61850 {
31       parameters:
```

```
32          driver = driver;
33          scl = scl;
34          ip = ip;
35      }
36      NetworkLayer: IEDNetwork {
37        tcpType = tcpType;
38        IPForward = IPForward;
39        namid = namid;
40        routingFile = routingFile;
41        numExtInterfaces = numExtInterfaces;
42        queueType = queueType;
43      }
44    connections allowunconnected:
45      IEC61850.mmsOut --> NetworkLayer.mmsAppIn++;
46      IEC61850.mmsIn <-- NetworkLayer.mmsAppOut++;
47      IEC61850.ethOut --> NetworkLayer.ethIn;
48      IEC61850.ethIn <-- NetworkLayer.ethOut;
```

Listing 5.2
IED NED module partial listing implementation.

5.4.2.2.1 Compound Modules The following compound modules are included in the Logical model, refer to figure 5.8 for a graphical representation.

IED The IED compound module represents the IED (secondary equipment). Inside it the modules with the Ethernet logic (IEDNetwork) and also the IEC 61850 logic and data model (LNs and LDs) are included.

IEDNetwork This compound module contains the modules from INET for Ethernet simulation with some small additions to handle MMS messages. This is currently incomplete code and it is not used to the current prototype tool since it is not needed for testing. In the future this code needs to be completed in order to completely simulate the substation.

LD The functional logic is included in the Logical Nodes (LNs) but these are grouped inside the Logical Devices (LDs). This was done in order to follow the IEC 61850 presentation and data modeling and to avoid having too many modules in the same hierarchy level, useful when providing graphical representation of the hierarchy.

5.4.2.2.2 Simple Modules The following simple modules were developed for the logic model.

Logical Node (LN) XSWI The XSWI interfaces directly with the switching primary equipment, i.e. CBR and DIS. The state of the switch is monitored using a pooling approach and during configuration the primary equipment name and location must be setup. Tables 5.3 and 5.4 list the module events and functionality description.

Table 5.3
Logical Node (LN) XSWI simple module events.

Event	Origin	Description
Block Open	CILO LN	Prevent further Switch opening operations.
Block Close	CILO LN	Prevent further Switch closing operations.
Open Switch	CSWI LN	Operate the switch to the open position.
Close Switch	CSWI LN	Operate the switch to the closed position.
Timeout	Self	Self-message event to model the timeout of the Select Before Operate behavior.
Pooling	Self	Self-message event to pool current Switch position.
Subscription	CSWI/CILO LNs	Receive notifications of Switch position changes.

Logical Node (LN) CSWI The CSWI interfaces directly with the XSWI LN. Tables 5.5 and 5.6 describe the module events and main functionality.

Logical Node (LN) CILO The CILO Logical Node (LN) implements the interlocking logic. In this initial model a simple logic of state changes of a specific switch based on the current state from another switch was implemented. Further future works must improve and develop further logic models. Tables 5.7 and 5.8 describe the events and main functionality of the module.

5.4.2.2.3 Others Other LNs from the standard were also implemented without any functionality also because of the previously mentioned reasons, lack of interest to the currently intended functionality and keeping the standard structure while avoiding unnecessary extra logics for class names conversion.

5.4.3 ”scl2ned” library

The objective of the ”scl2ned” library is to read the Substation Configuration Description (SCD) file and to generate the NED topology files for the simulation engine. For this the file needs to be

Table 5.4
Logical Node (LN) XSWI simple module functionality.

Action/Event	Description
Blocking	When the block command is received, for open and for closing, the XSWI LN is put into a state that prevents any further, open or close, commands to be executed. This command accepts a boolean value for enabling and disabling this feature.
Operation	The operation is dependent on the control model used, see section 3.5.1.1. This control model is described, *per* LN, in the SCL file as it depends on configuration and setup. When an operate message to either open or close the switch is received several conditions must first be verified. First the current state of the switch must be verified, if the switch is in a failure state or already in the pretended state then no action is executed. Also if the command is to open the switch and the block open event has been previously issued then no opening of the switch is possible. Likewise if the command is to close the switch and the block close event has been previously issued then again no action is possible.
Selection	Depending on the control model used, before any operation the device data must first be selected in order for the command to be accepted.
Timeout	When the device data is selected the operation command must be issued within a specific amount of time. If this timeout occurs and no command has been issued then the device reverts back to the unselected state.
Pooling	By default, every 1 millisecond, the XSWI LN queries the current state of the switch. If a change between the previous state and the current state is detected all subscribed LNs receive an update event.
Subscribe	Other LNs can subscribe to receive update event notifications whenever a change of state occurs in the switch.

Table 5.5
Logical Node (LN) CSWI simple module events.

Event	Origin	Description
Open Switch	HMI	Operate the switch to the open position.
Close Switch	HMI	Operate the switch to the closed position.
Timeout	Self	Self-message event to model the timeout of the Select Before Operate behavior.
Update	XSWI LN	Change of state occurred in the subscribed LNs.

read using a XML parser while validating against the SCL schema for edition 1 of the standard. This builds a tree model, in memory, of everything described in the standard (substation, devices (IEDs), communication network and connections). From this model one can later generate all

Table 5.6
Logical Node (LN) CSWI simple module functionality.

Action/Event	Description
Configuration	During initialization and configuration the CSWI LN subscribes to the connected XSWI LN in order to receive updated notifications whenever the switch changes state.
Operation	The operation is dependent on the control model used, see section 3.5.1.1. This control model is described, *per* LN, in the SCL file as it depends on configuration and setup. The command is relayed down to the XSWI in order to operate the connected switch.
Selection	Depending on the control model used, before any operation the device data must first be selected in order for the command to be accepted.
Timeout	When the device data is selected the operation command must be issued within a specific amount of time. If this timeout occurs and no command has been issued then the device reverts back to the unselected state.
Update	When the CSWI is subscribed to a XSWI LN and the switch changes state the update event notification is received.

Table 5.7
Logical Node (LN) CILO simple module events.

Event	Origin	Description
Update	XSWI LN	Change of state occurred in the subscribed LNs.

Table 5.8
Logical Node (LN) CILO simple module functionality.

Action/Event	Description
Configuration	During initialization and configuration the CILO LN subscribes to the *monitored* Switch.
Update	Whenever the *monitored* Switch changes state the connect (or controlled) Switch is either blocked or unblocked depending on the logic implemented.

the necessary files by using different classes named *serializers* that traverse the tree while directing the output to a stream (file). The different needed files are the NED topology files, the INI and the MRT files. Development of this library was done using C++/CLI language. This allows access to the powerful .NET® framework and provides a quick development environment.

5.4.3.1 NED *serializer*

The NED topology files are built by reusing the available modules from *EPS* framework. Here several inputs from the user are necessary. These inputs include a selection of the driver DLL used by each of the *IEC61850* modules, the network interface to use (if there is more than one available) and input on how to serialize the substation. This input on how to serialize the substation refers to the dynamic logic introduced in order to consider different levels of substation as NED modules or not. This became a necessity after realizing that, except for the most simple substations, the graphical representation becomes too complex to be useful without modularization. Initially the default graphical model representation from OMNeT++ was being used, this might also be a useful feature in the future for a simulation only or educational tool. Also necessary is the identification of the connection nodes that are electrical grounds and also busbars. This identification can more or less be automated by smart analysis of the substation model but may also require user input. This allows the busbars graphical representation to be replaced by a bus and the ground terminals to be split and to add a ground graphical representation. This simplifies the graphical representation and avoids many overlapping connections. The main idea is to bring the graphical representation closer to that of the more familiar single line diagram present in most SCADA terminals. These two features already improve in this sense but further developments need to be done. Since the GUI development was dropped in favor of a faster and simpler to develop one (without graphical representation of the simulation model), no further work was done in this direction.

In reality two different types of *IED* modules were developed. One named *DRIVER* contains the driver functionality described above. The user is required to select a driver to be loaded by the model. The other one, named *CLIENT*, refers to a *IED* module that contains no functionality and hence does not require a driver. This necessity comes from the fact that several substations contain many devices that are not relevant to the use of IEC 61850 standard. Occasionally one may also require the simulation of certain devices within the substation but not all. This allows more control over the simulation environment and also on the resources required by it. The NED *serializer* takes advantage of the inheritance properties provided by the NED language and during serialization these different *IED* modules are built by subclassing the main *IED* module.

5.4.3.2 INI *serializer*

The INI *serializer* is quite simple so far and just needs to account for some parameters that configure real interfaces and global simulation configuration options. This configuration is related to the *WinPCap* library and refers to the filter and network interface to use for each of the mod-

els. The global configuration options include the type of scheduler class used, the simulation time granularity and output related options. In the future this will be expanded to accommodate further options and the code was implemented having this in mind.

5.4.3.3 MRT *serializer*

The INET framework interfaces can be configured by the use of a text file with a specific syntax. The syntax itself resembles the network configuration on a Linux machine. This text file ends with the MRT extension. The job of the MRT *serializer* is to read the communication and devices model from the SCD and to generate these files automatically. Later the simulation engine will use them to configure the models interfaces.

5.4.4 Driver *kemaieddrv*

Based upon the API developed for the *IEC61850* module, the driver named *kemaieddrv* was developed. This driver however implements an hybrid model. The driver provides access to the IEC 61850 communication stack but the logic remains in the logic model. This decision is based on time constraints. KEMA currently provides the UniCA Multi-IED simulator. This tool is already capable of simulating multiple IED devices. These devices comply to the IEC 61850 standard but do not provide any logic functionality. So the logic model is extended to include the logic functionality, normally expected to be inside the driver.

This allows the quick development of the *IED simulator* without requiring to build the IEC 61850 communication stack.

5.4.5 User interface

Initial tests and development were done using the user interface provided by OMNeT++ based on TCL/TK technology. Soon this option was dropped in favor of an embedded user interface. This allows for a faster and better integration of the GUI with the simulation framework. Following the recommendations in the OMNeT++ manual, the work was initially done, by copying Cmdenv and adding features and functionality as necessary. This involved the creation of a library named *EPSEnv* in C++ and a wrapper library named *EPSEnvWrapper* in C++/CLI. This wrapper is necessary so that we are able to integrate it into the C# GUI. During execution the simulation engine sends periodic reports to *EPSEnv* that contain several statistics about simulation execution along with console messages that are translated into a log window in the GUI. This allows the user to monitor the initialization, execution and finalization of the simulation in case of some unexpected errors occur. The GUI also provides a button that allows the user to

cancel and stop the currently running simulation. It was also necessary to develop in the GUI a configuration menu that allows the user to provide the input to the *scl2ned* library in an easy to use way. This menu presents to the user, after loading the SCL file, the complete loaded model of the substation, devices and communication parts. Information was organized into different tabs in order to better separate the different models and its configuration. There is also a log window in the configuration menu. This window allows the presentation to the user of warnings and errors found during the loading of the SCL due to nonconformities of the schema. In some cases these warnings and errors do not influence the outcome of the results, as when specific companies do not follow the standard strictly, and therefore must be evaluated by the user.

Chapter 6

Results

6.1 Testing tools and platforms

The testing environment was composed of a DELL OPTIPLEX® GX620 desktop computer running Windows XP® Professional with Service Pack 3. This machine would host a VMWare® image, workstation 6.0 virtual machine version, of the same operating system version. In order to provide some validity to the results the tool provided by KEMA, UniCASim 61850 Client Simulator, would be used to issue commands to the devices in the simulation environment. This tool is currently used in the Industry to test and certificate devices according to IEC 61850 standard.

This setup was chosen for several reasons. No real Intelligent Electronic Devices (IEDs) were available for testing. Also the currently developed *IED simulator*, provided by *kemaieddrv* library and the logic model, do not have Generic Object Oriented Substation Events (GOOSE) and reporting implemented yet. This makes it impossible, so far, to interact totally with real devices. The usage of the UniCASim 61860 Client Simulator already proves that the already developed part behaves correctly. Another reason is the fact the *scheduler* do not have yet a scripting engine developed. The commands could have been issued directly by the scheduler, *hardcoded* and without the manual work from UniCASim 61850 Client Simulator, behaving like the test script. It was however decided that this would bring no real value to the test and the setup used was more flexible. Changes could be made to the test, like a test script would, without the need to recompile the project.

6.2 Testing setup and environment

The guest operating system would run the simulator in a closed private network shared with the host (host-only in VMWARE® *jargon*). The host operating system would run the UniCASim 61850 Client Simulator, referred from now on only as the client simulator.

For the test of the simulator the substation ”MAGNUM IGCC POWER PLANT 380kV SAS” was chosen. This substation is already large containing a completely and correctly described Substation Configuration Description (SCD) file, with approximately 1.633 kB, that is correctly read and parsed by the ”scl2ned” library. This substation contains 10 Bays and 29 devices split among 4 subnetworks. In figure 6.1 part of the substation description, taken from the SCD file, is represented graphically in a single-line diagram. For the tests developed and

described in this work the equipment chosen was the disconnectors QC9 and QC1 from bay Q04.

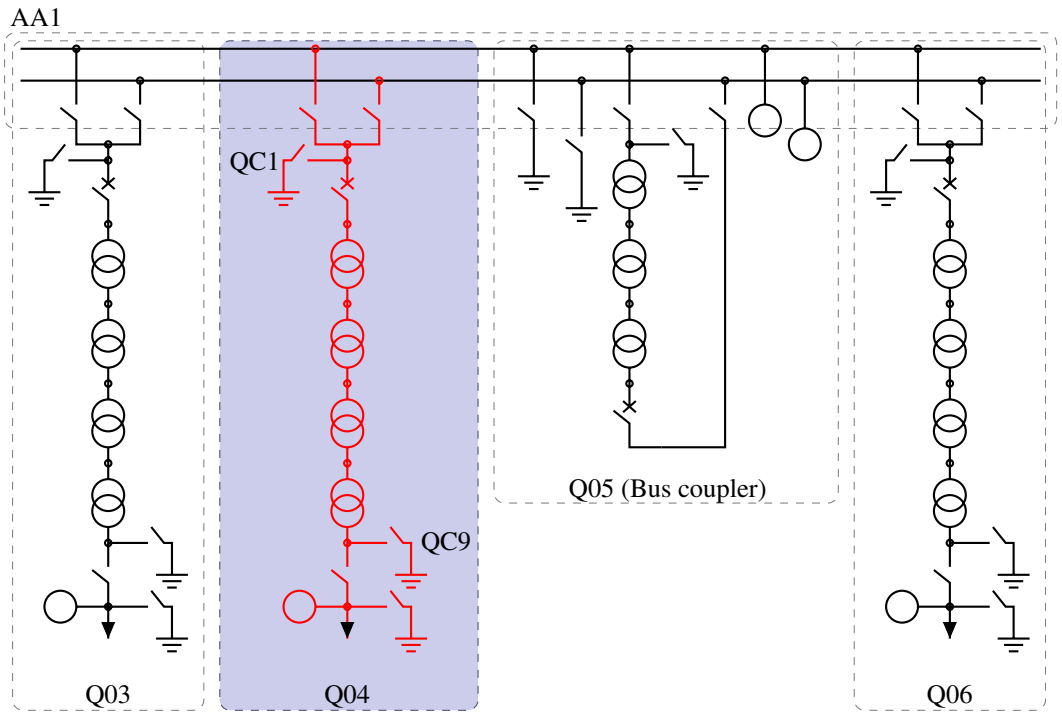

Figure 6.1
Partial single-line diagram for MAGNUM AA1 substation. The test scenario focus on disconnectors QC1 and QC9 from Q04 bay. The complete substation contains 10 bays.

The devices and logical devices that are related to these equipment, referring from the SCD file, are *AA1C1Q04A1* and *LD0* respectively. Referenced, according to the standard, as *AA1C1Q04A1/LD0*. Specifically Logical Nodes (LNs) SXSWI7 and SXSWI5 are connected to disconnectors QC9 and QC1 respectively. Configuration and setup of the remaining LNs are closely related to the logics modeling that is still missing in the standard. Since the logic modeling still needs to be further investigated and developed the following setup was decided for this initial test, for no particular reason. LNs SCSWI7 and SCSWI5 were connected to LNs SXSWI7 and SXSWI5 respectively. LN SCILO7 would control the interlocking allowing the operation of disconnector QC9 as long as disconnector QC1 would be in the opposite position (only *on* and *off* states were considered). The scheduler was configured to terminate successfully when disconnector QC9 would achieve the state *on* while disconnector QC1 was also on the *on* state and unsuccessfully otherwise. Both disconnectors would start in the *off* state. Refer to table 6.1. Both SCSWI7 and SCSWI5 *pos* DATA were operating with Select Before Operate (SBO) control model, according to the configuration options from the SCD file.

Table 6.1
Ending conditions specified for the test scenario. The *scheduler* would monitor both devices and decide upon the conditions how to flag the test.

QC9 state	QC1 state	*scheduler* condition
off	*off*	continue execution (initial condition)
off	*on*	continue execution
on	*off*	terminate condition failure
on	*on*	terminate condition success

Figure 6.2 shows a screen shot of the simulator taken during the configuration step of the devices.

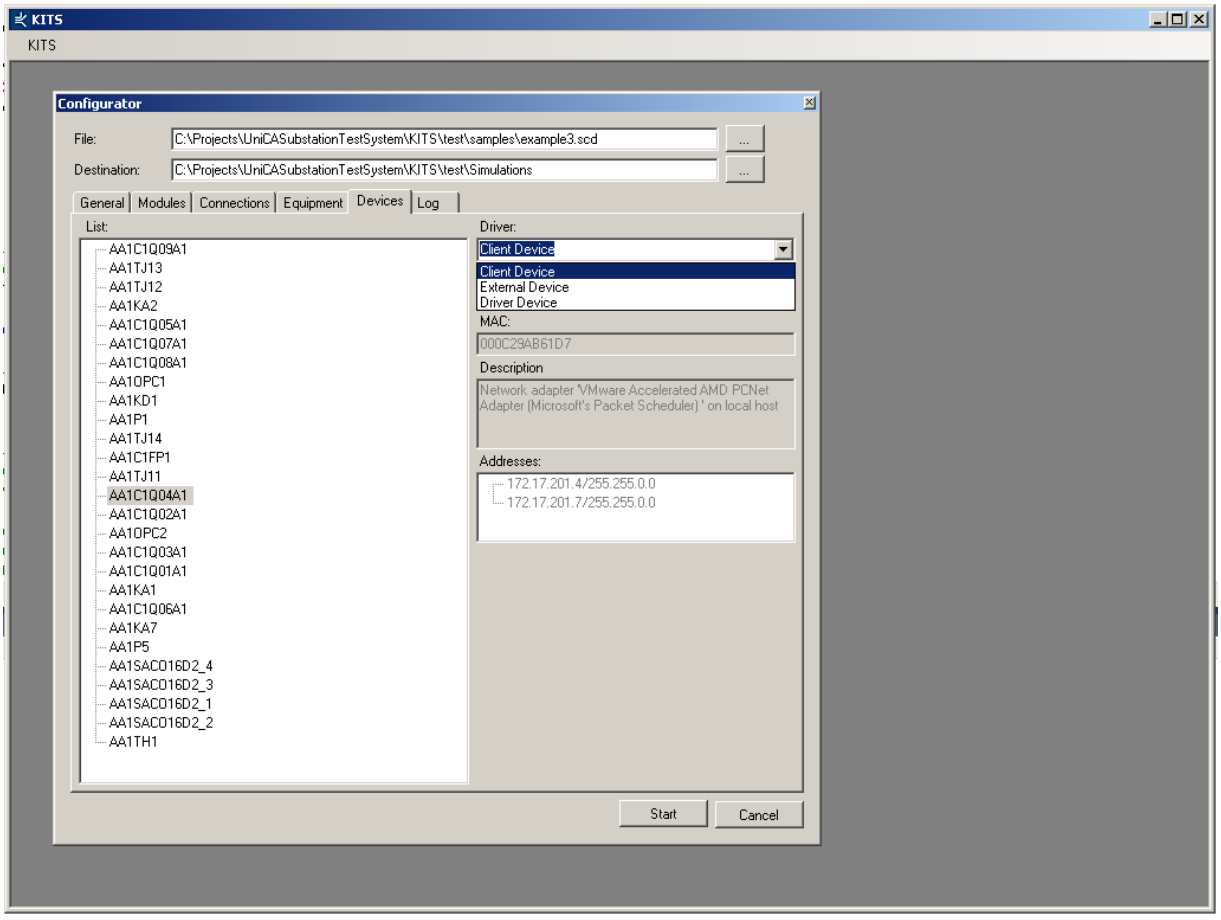

Figure 6.2
Screen shot of the configuration menu while selecting the driver for device *AA1C1Q04A1*.

6.3 Execution

After configuring and setting up the simulation, according to the previous description, the test was executed and a log file was generated. This log (ELOG) file is examined in detail and discussed in this section. Figure 6.3 shows a simplified graphical overview of the configuration and setup used.

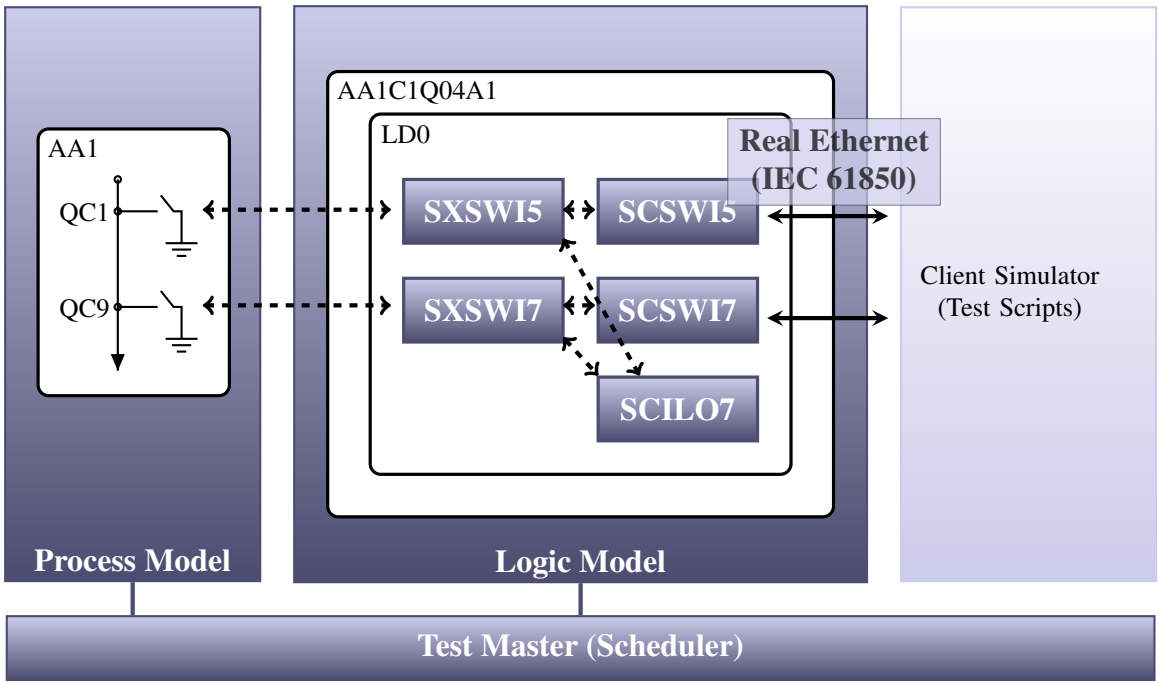

Figure 6.3
Simplified graphical overview of the test scenario configuration and setup used.

All the required LNs initialized and configured correctly in the simulation environment. Listing 6.1 shows the relevant parts of the log regarding LNs initialization. All the initialization happens at the predetermined time of 2 seconds (simulation time). This accounts for a default, but configurable, 2 seconds for device boot up. SCSWI7 and SCSWI5 both contain a Common Data Class (CDC) *Pos* with a control model SBO with enhanced security. Both SXSWI7 and SXSWI5 contain CDC *BlkCls*, *BlkOpn* and *BlkUpd* using direct with normal security control models and also a *Pos* with a status only control model. Information summarized in table 6.2.

```
12051 E # 113 t 2 m 565 ce 0 msg 507
12052 BS id 795 tid 795 eid 795 etid 795 c cMessage n "AA1C1Q04A1LD0/SCILO7 Configure
      event"
12053 ES t 3
12054 - We are behind: 4.99 seconds

12072 E # 118 t 2 m 570 ce 0 msg 512
12073 - AA1C1Q04A1LD0/SCSWI5: Pos CtlModel sbo-with-enhanced-security
12074 BS id 815 tid 815 eid 815 etid 815 c cMessage n "AA1C1Q04A1LD0/SCSWI5 Configure
      event"
12075 ES t 3
12076 - We are behind: 5.01 seconds

12082 E # 120 t 2 m 572 ce 0 msg 514
12083 - AA1C1Q04A1LD0/SCSWI7: Pos CtlModel sbo-with-enhanced-security
12084 BS id 823 tid 823 eid 823 etid 823 c cMessage n "AA1C1Q04A1LD0/SCSWI7 Configure
      event"
12085 ES t 3
12086 - We are behind: 5.01 seconds

12188 E # 146 t 2 m 598 ce 0 msg 540
12189 - AA1C1Q04A1LD0/SXSWI5: BlkCls CtlModel direct-with-normal-security
12190 - AA1C1Q04A1LD0/SXSWI5: BlkOpn CtlModel direct-with-normal-security
12191 - AA1C1Q04A1LD0/SXSWI5: BlkUpd CtlModel direct-with-normal-security
12192 - AA1C1Q04A1LD0/SXSWI5: Pos CtlModel status-only
12193 BS id 927 tid 927 eid 927 etid 927 c cMessage n "AA1C1Q04A1LD0/SXSWI5 Configure
      event"
12194 ES t 3
12195 - We are behind: 5.02 seconds
12196
12197 E # 147 t 2 m 599 ce 0 msg 541
12198 - AA1C1Q04A1LD0/SXSWI6: iec61850 module not configured, sleeping!!!
12199 - We are behind: 5.02 seconds
12200
12201 E # 148 t 2 m 600 ce 0 msg 542
12202 - AA1C1Q04A1LD0/SXSWI7: BlkCls CtlModel direct-with-normal-security
12203 - AA1C1Q04A1LD0/SXSWI7: BlkOpn CtlModel direct-with-normal-security
12204 - AA1C1Q04A1LD0/SXSWI7: BlkUpd CtlModel direct-with-normal-security
12205 - AA1C1Q04A1LD0/SXSWI7: Pos CtlModel status-only
12206 BS id 935 tid 935 eid 935 etid 935 c cMessage n "AA1C1Q04A1LD0/SXSWI7 Configure
      event"
12207 ES t 3
12208 - We are behind: 5.02 seconds
```

Listing 6.1
Initialization of devices logical nodes

Initially one attempt to activate the QC9 disconnector was made using the client simulator to connect and operate to SCSWI7 by following the correct procedures for SBO control model. The command was successfully denied and QC1 remained in the *off* position. Listing 6.2 shows the relevant events of the log file. Since SCSWI7 is working on SBO mode, one must first

Table 6.2
LNs CDCs Control Models as loaded and configured from the SCD file into the simulation.

	Control Model (CDC)			
LN	*Pos*	*BlkCls*	*BlkOpn*	*BlkUpd*
SCSWI5	SBO with enhanced security	–	–	–
SCSWI7	SBO with enhanced security	–	–	–
SXSWI5	status only	direct, normal	direct, normal	direct, normal
SXSWI7	status only	direct, normal	direct, normal	direct, normal

select the object before executing the command. The *Select* command is accepted correctly at time 14.12 seconds (simulation time) as can be observed by the change of the state to *Ready* state. listing line number 316190. At time 15.287 seconds, approximately one second later, the *Operate* command is sent and fails due to a block closing restriction in SXSWI7 controlled by SCILO7. listing line number 351202. The long time between the *Select* and the *Operate* commands is due to the fact that the client simulator is being executed manually.

```
316187 E # 50838 t 14.12 m 572 ce 50833 msg 822
316188 BS id 825 tid 825 eid 825 etid 825 c cMessage n "AA1C1Q04A1LD0/SCSWI7 Timeout event"
316189 ES t 44.12
316190 – AA1C1Q04A1LD0/SCSWI7: Pos state changed to Ready
316191 BS id 822 tid 822 eid 822 etid 822 c cMessage n "AA1C1Q04A1LD0/SCSWI7 Pooling event"
           pe 50838
316192 ES t 14.121
316193 – We are behind: 53.568 seconds

351200 E # 56673 t 15.287 m 572 ce 56668 msg 822
351201 CE id 825 pe 50838
351202 – AA1C1Q04A1LD0/SXSWI7: Switch Operation blocked due to BlkCls!
351203 – AA1C1Q04A1LD0/SCSWI7: Pos state changed to Unselected (Operate FAILED)
351204 BS id 822 tid 822 eid 822 etid 822 c cMessage n "AA1C1Q04A1LD0/SCSWI7 Pooling event"
           pe 56673
351205 ES t 15.288
351206 – We are behind: 59.081 seconds
```

Listing 6.2
Attempt at operation of disconnector QC9 while QC1 is in forbidden state.

Then QC1 disconnector was changed state by operating SCSWI5 following also the SBO control model. The command was successfully interpreted and the disconnector was successfully operated. Listing 6.3 shows the relevant parts of the log file. Just like previously, the object must first be selected before operation. The difference is that the operate is accepted successfully, as observed by the change of state to *WaitForChange* at time 18.094 seconds. Listing line number 435419. After the command propagates downwards from SCSWI5 to SXSWI5

and finally to QC1 equipment, the switch initiates the requested action and finally achieves the new steady state 10 milliseconds later (default time, also configurable). This change of state is then noticed by SXSWI5 that propagates the update upwards until it finally reaches SCSWI5 and triggers a change to *Unselected* state at 18.104 seconds. There is no communication delay because they both reside in the same device and SXSWI5 is again ready to accept another *Select* command.

```
406458 E # 65882 t 17.129 m 570 ce 65877 msg 814
406459 BS id 817 tid 817 eid 817 etid 817 c cMessage n "AA1C1Q04A1LD0/SCSWI5␣Timeout␣event"
406460 ES t 47.129
406461 – AA1C1Q04A1LD0/SCSWI5: Pos state changed to Ready
406462 BS id 814 tid 814 eid 814 etid 814 c cMessage n "AA1C1Q04A1LD0/SCSWI5␣Pooling␣event"
           pe 65882
406463 ES t 17.13
406464 – We are behind: 67.544 seconds

435411 E # 70707 t 18.094 m 570 ce 70702 msg 814
435412 CE id 817 pe 65882
435413 BS id 70 tid 70 eid 70 etid 70 c cMessage n "Switch␣close␣Event"
435414 ES t 18.104
435415 BS id 71 tid 71 eid 71 etid 71 c cMessage n "Switch␣timeout␣Event"
435416 ES t 18.109
435417 BS id 817 tid 817 eid 817 etid 817 c cMessage n "AA1C1Q04A1LD0/SCSWI5␣Timeout␣event"
           pe 70707
435418 ES t 48.094
435419 – AA1C1Q04A1LD0/SCSWI5: Pos state changed to WaitForChange (Operate)
435420 BS id 814 tid 814 eid 814 etid 814 c cMessage n "AA1C1Q04A1LD0/SCSWI5␣Pooling␣event"
           pe 70707
435421 ES t 18.095
435422 – We are behind: 72.148 seconds

435751 E # 70762 t 18.104 m 570 ce 70761 msg 816
435752 – AA1C1Q04A1LD0/SCSWI5: Switch change detected!
435753 – AA1C1Q04A1LD0/SCSWI5: Pos state changed to Unselected (updated)
435754 – We are behind: 72.228 seconds
```

Listing 6.3
Operation of disconnector QC1 in order to enable the successful operation of QC9.

After this step another try at operating disconnector QC9 was made and this time, as expected, it was successfully accepted and the switch changed state. Listing 6.4 shows the relevant events of the log. Please note the switch change detected message is not shown. This is due to the fact that the scheduler "catches" the termination condition before the successful state change can be reported back to SCSWI7 LN.

```
474414 E # 77206 t 19.393 m 572 ce 77201 msg 822
474415 BS id 825 tid 825 eid 825 etid 825 c cMessage n "AA1C1Q04A1LD0/SCSWI7␣Timeout␣event"
           pe 56673
474416 ES t 49.393
```

```
474417 – AA1C1Q04A1LD0/SCSWI7: Pos state changed to Ready
474418 BS id 822 tid 822 eid 822 etid 822 c cMessage n "AA1C1Q04A1LD0/SCSWI7␣Pooling␣event"
           pe 77206
474419 ES t 19.394
474420 – We are behind: 78.06 seconds

507057 E # 82646 t 20.481 m 572 ce 82641 msg 822
507058 CE id 825 pe 77206
507059 BS id 64 tid 64 eid 64 etid 64 c cMessage n "Switch␣close␣Event"
507060 ES t 20.491
507061 BS id 65 tid 65 eid 65 etid 65 c cMessage n "Switch␣timeout␣Event"
507062 ES t 20.496
507063 BS id 825 tid 825 eid 825 etid 825 c cMessage n "AA1C1Q04A1LD0/SCSWI7␣Timeout␣event"
           pe 82646
507064 ES t 50.481
507065 – AA1C1Q04A1LD0/SCSWI7: Pos state changed to WaitForChange (Operate)
507066 BS id 822 tid 822 eid 822 etid 822 c cMessage n "AA1C1Q04A1LD0/SCSWI7␣Pooling␣event"
           pe 82646
507067 ES t 20.482
507068 – We are behind: 83.241 seconds
```

Listing 6.4
Successful operation of disconnector QC9.

The scheduler detects successfully the successful termination condition and ended the simulation execution with the success flag. Listing 6.5 shows the relevant events of the log where at time 20.491 seconds the switch is found to be closed (*on*) and the simulation terminates.

```
507354 E # 82694 t 20.491 m 72 ce 82646 msg 64
507355 CE id 65 pe 82646
507356 – Switch: closed (ON)
507357 – Objective successfull

507397 – NET_AA1.AA1C1Q04A1.NetworkLayer.tcp: finishing with 0 connections open.
507398 – NET_AA1.AA1C1Q04A1.NetworkLayer.sctp: finishing SCTP with 0 connections open.
507399 – [20.491] tcpdump finished
```

Listing 6.5
The scheduler detects the success ending condition and ends the simulation with the success flag.

The current code uses a pooling approach to detect changes in the LNs data. This approach was chosen due to limitations in the currently used code from UniCA Sim Client Simulator. The pooling messages are being generated by default at a 1 ms rate but these are configurable parameters. In the previous simulation it can be observed that the simulation clock is running behind the wall clock (real time) and ends with more than one minute of difference. Although the compiled and executed code was not optimized, which could have improved this, this may prove detrimental to the required time stamp accuracy of 1 ms when simulating many devices. Changing the used approach into a function callback system (signals) will generate less *traffic* in

the *scheduler* and improve the resources usage of the test system. Figure 6.4 shows a screen shot of the simulator program during execution. Several statistics like real elapsed time, simulation elapsed time, number of events, messages created and others are updated regularly to provide some feedback to the user besides the log window.

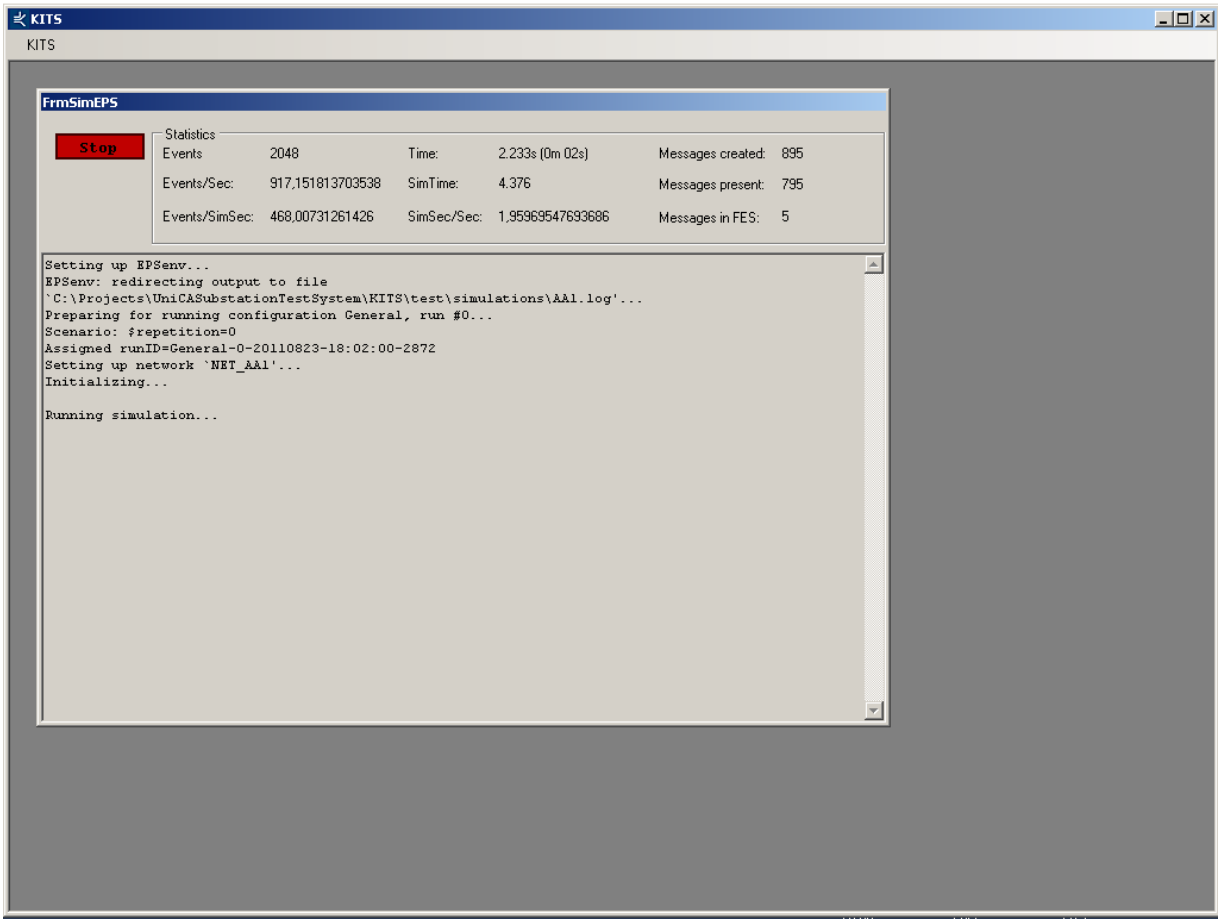

Figure 6.4
Screen shot of the simulator during execution. The performance statistics is updated regularly.

After the test, the simulation processed 82694 events and simulated 20.491 seconds in 103 seconds and 801 milliseconds.

Chapter 7

Conclusions

The previously described test scenario executes successfully and as expected. The parts that are already developed are working correctly. The *IED simulator* already presents itself as a real Intelligent Electronic Device (IED) to external real devices. The functionality with itself and other virtual devices, not demonstrated in the test scenario, is already functional but somewhat limited. This limitation is mainly in the logic configuration of the devices which is partly *hardcoded*. Further developments are necessary in this area taking into consideration future extensions in the standard that account for logic modeling. The complete functionality with real devices cannot be achieved before the reporting and the Generic Object Oriented Substation Events (GOOSE) are completely implemented.

For all these reasons we consider that the objectives set for this study were successfully achieved even if not totally. In order to become a successful and complete application several features need to be improved and others to be added. Further time and investment will allow a complete solution able to be commercialized. This project is part of a study program and as such it had a reduced amount of limited resources. While figure 5.4 presents the vision, figure 7.1 sets the goals and figure 6.3 summarizes the achievements.

Observing the intended goals of the test system, figure 7.1, we can summarize the achievements of the project. The Logic Model requires a further development to include GOOSE, reporting and logic modeling (configuration of devices). The Test Scripts engine needs to be implemented, maybe by reusing code or parts of the code from the current UniCA Multi-IED Simulator. The current I/O Systems in the market need to be studied and I/O Drivers for the most useful ones need to be developed. All other objectives were met. Once this is done testing of real IEDs can be achieved. The next section will detail some of the future work that needs, some optionally, to be developed in order to improve the tool and allow it to become production ready.

7.1 Future work

Further develop the driver approach to the different modules for substation primary equipment (I/O Driver System). This will allow the connection and interface of external equipment with the simulation environment enabling the complete integration of real IEDs with the simulation environment.

Substation functionality is generally the same across different substations but small imple-

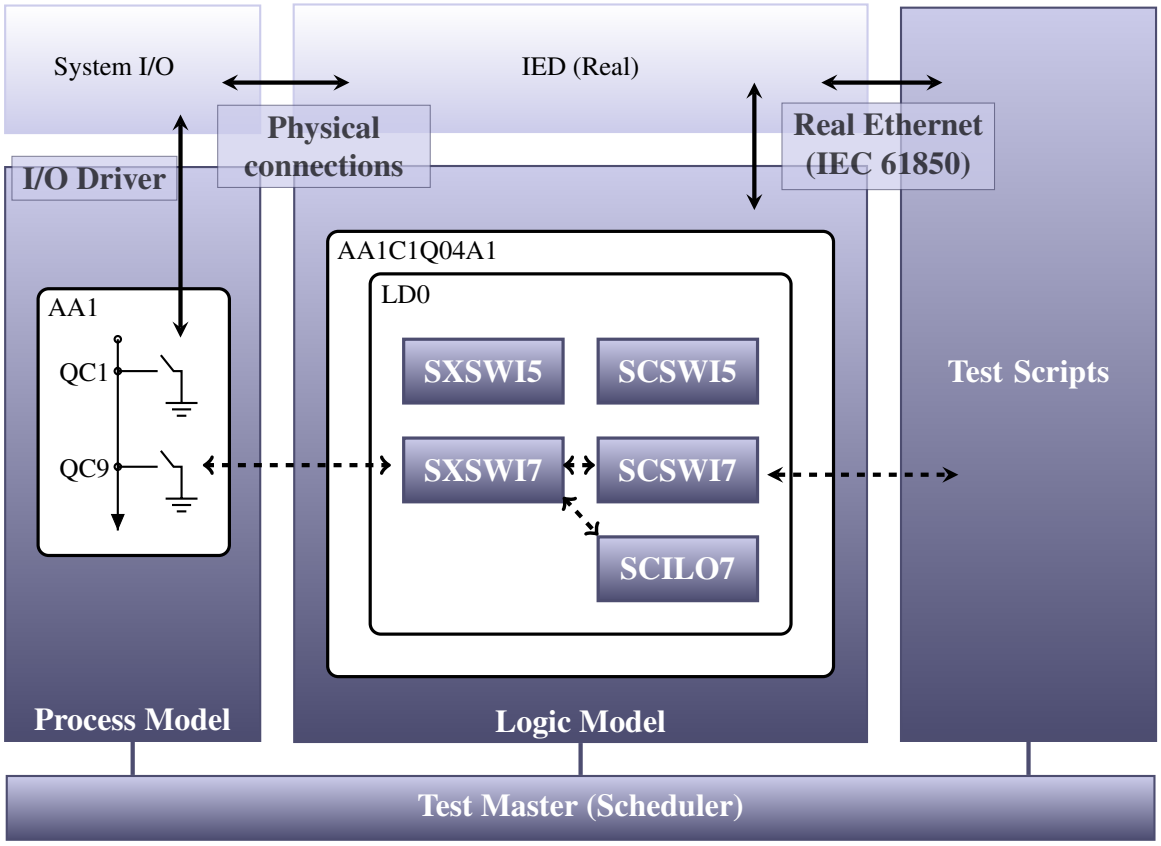

Figure 7.1
Goals of the test system. This example application of the complete test system demonstrates a possible test scenario where part of the previous test logic resides in a real IED to be tested, i.e. the previous functionality of SXSWI5 and SCSWI5 is now performed by the real IED.

mentation details may arise. In order to allow a quick and easy adaption of the testing conditions to the specific implementations it is necessary to develop a scripting language and integrate a scripting engine into the current scheduler.

Develop a reporting tool in the Graphical User Interface (GUI) to allow quick and easy assessment, analyzes and processing of test results.

Currently there is no much output from the simulator except for some statistical performance numbers and a log window. If it proves to be worthwhile then the GUI needs to be changed in order to represent the single-line diagram of the substation under test. This is already partly possible but further improvements are necessary. This requires further development of the embedded GUI and the replacement of the current layout algorithm for module placement. Being able to represent the substation graphically in a way similar to existing Supervisory Control And Data Acquisition (SCADA) products could enable a faster training and familiarity with the tool for the technical personnel.

A study regarding the usefulness of a tool for complete simulation of a substation, without real devices, and/or Energy Power System (EPS) system (complete or partial grid) can be executed. This can be useful for prediction and analyzes of different grid, substation and/or Substation Automation System (SAS) architectures or topologies before commitment. For that the current system can easily be expanded to a complete simulation tool. For this goal the following developments are necessary. Develop the Process Model to simulate electrical power flow. Further improve the Logic Model and develop the remaining functionality, since currently only switching and interlocking is possible.

Edition 2 of the IEC 61850 standard is about to be released and this will certainly bring some changes that will require to be taken into account.

Currently the *IED Simulator* relies on the network stack provided by KEMA UniCA Multi-IED simulator. This makes the communication between devices to be out of the control of the simulation environment. In order to change this several developments are necessary one of them being the expansion of the INET/*EPS* framework to understand Manufacturing Message Specification (MMS).

IEC 61850 standard is just one of the latest developments that are currently being introduced inside the substation but the complete EPS universe is very vast and rich. There are currently many standards and protocols used beside IEC 61850 and it will take some time before IEC 61850 is fully absorbed by the market. This means that there is the need and also the room for expansion in the simulator, both vertical and horizontal. Horizontal means the addition of several other protocols used inside the substation while vertical means the expansion of the simulator to allow the simulation between different substations and possibly the entire grid.

REFERENCES

Albrecht, M. (2010). Introduction to discrete event simulation.

Baigent, D., Adamiak, M., and Mackiewicz, R. (2005). IEC 61850 Communication Networks and Systems in Substations: An overview for users.

Brand, K., Brunner, C., and Wimmer, W. (2004). Design of iec based substation automation systems according to customer requirements. Technical report, ABB Switzerland Ltd, Baden and Zurich.

Brand, K.-P. (2004). The Standard IEC 61850 as Prerequisite for Intelligent Applications in Substations. *Power Engineering Society General Meeting, 2004. IEEE*, 2.

Brand, K.-P., Lohmann, V., and Wimmer, W. (2003). *Substation Automation Handbook*. Utility Automation Consulting Lohmann, Im Meyerhof 18, CH-5620 Bremgarten, Switzerland.

Dubuisson, O. and Fouquart, P. (2000). ASN.1 Communication between Heterogeneous Systems.

Haffar, M., Thiriet, J. M., and El-Nachar, M. (2010). Software and hardware in the loop component for an IEC 61850 co-simulation platform. *Proceedings of the International Multiconference on Computer Science and Informaiton Technology*, pages 817–823.

IEC 61346-1 (1996). Part 1: Basic rules. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-1 (2003). Part 1: Introduction and overview. Technical report, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-2 (2003). Part 2: Glossary. Technical specification, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-3 (2002). Part 3: General requirements. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-4 (2002). Part 4: System and project management. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-5 (2003). Part 5: Communication requirements for function and device models. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-6 (2004). Part 6: Configuration description language for communication in electrical substations related to IEDs. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-7-1 (2003). Part 7-1: Basic communication structure for substation and feeder equipment – principles and models. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-7-2 (2003). Part 7-2: Basic communication structure for substation and feeder equipment – abstract communication service interface (ACSI). International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-7-3 (2003). Part 7-3: Basic communication structure for substation and feeder equipment – common data classes. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-7-4 (2003). Part 7-4: Basic communication structure for substation and feeder equipment – compatible logical node classes and data classes. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-8-1 (2004). Part 8-1: Specific communication service mapping (SCSM) – mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 61850-9-1 (2003). Part 9-1: Specific communication service mapping (SCSM) – sampled values over serial unidirectional multidrop point to point link. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

IEC 81346-1 (2009). Part 1: Basic rules. International standard, International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland.

ISO 8802-3 (2000). Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. International standard, International Organization for Standardization, Case postale 56, CH-1211 Geneva 20, Switzerland.

Marshall T. Rose and Dwight E. Cass (1987). RFC 1006: ISO Transport Service on top of the TCP. Standard, Northrop Research and Technology Center.

Moreno, J. G. (2010). Modeling of logics use cases analysis. Technical report, IBERDROLA, TORONTO 14th, Rev.1.

OMNeT++ Community (2011). OMNeT++ Community welcome. http://www.omnetpp.org/. Accessed: 26 of September, 2011.

Schimmel, R. (2008). Automated acceptance testing of IEC 61850: Optimise the testing from utility point of view. *Praxis Profiline*, pages 2–4.

Simulcraft, Inc. (2011). OMNEST - OMNeT++ comparison. http://www.omnest.com/comparison.php. Accessed: 26 of September, 2011.

Udren, E. A. and Dolezilek, D. (2006). IEC 61850: Role of Conformance Testing in Successful Integration. Proceedings of the Eighth Annual Western Power Delivery Automation.

Varga, A. (2010). *OMNeT++ User Manual Version 4.1*.

Varga, A. and Hornig, R. (2008). An overview of the OMNeT++ simulation environment. Marseille, France. Proceedings of SIMUTools 2008.

Wikipedia (2011). Wikipedia file for electricity grid in the US. http://en.wikipedia.org/wiki/File:Electricity_grid_simple-_North_America.svg. Accessed: September, 2011.